UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE
DEPARTAMENTO DE COMPUTAÇÃO

DEIVISON MOZER DE SOUZA

META-HEURÍSTICA CLUSTERING SEARCH PARA RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE JOGOS

ALEGRE - ES
2016

DEIVISON MOZER DE SOUZA

META-HEURÍSTICA CLUSTERING SEARCH PARA RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE JOGOS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.
Orientador: Prof. Dr. Geraldo Regis Mauri.

ALEGRE - ES
2016

DEIVISON MOZER DE SOUZA

META-HEURÍSTICA CLUSTERING SEARCH PARA RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE JOGOS

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em ____ de junho de 2016.

COMISSÃO EXAMINADORA

_____

Prof. Dr. Geraldo Regis Mauri
Universidade Federal do Espírito Santo
Orientador

_____

Prof. M.Sc. Antonio Almeida de Barros Junior
Universidade Federal do Espírito Santo

_____

Prof. M.Sc. Edmar Hell Kampke
Universidade Federal do Espírito Santo

DEDICATÓRIA

“Treine enquanto eles dormem, estude enquanto eles se divertem, persista enquanto eles descansam, e então, viva o que eles sonham.”

Provérbio japonês

AGRADECIMENTOS

Agradeço primeiramente a Deus, por todas as oportunidades e por ter me guiado até aqui.

À minha família, por todo o esforço em me manter estudando todos esses anos, pelo apoio e incentivo.

À minha namorada Suzanny Oliveira Mendes que sempre esteve ao meu lado durante toda a minha jornada.

Ao meu orientador Geraldo Regis Mauri pela oportunidade em desenvolver um projeto de iniciação científica que resultou neste trabalho, pela disponibilidade em ajudar, pela paciência e por todas as contribuições e direcionamentos.

À Universidade Federal do Espírito Santo e todos os seus funcionários, principalmente os professores do Departamento de Computação, que puderam transmitir seus conhecimentos.

À empresa Click Junior pela oportunidade de testar meus conhecimentos e me dar uma oportunidade de conhecer como funciona o mercado de trabalho.

Aos meus amigos que de alguma forma contribuíram com a minha formação.

RESUMO

Ao programarem competições, as ligas de confederações esportivas enfrentam uma grande dificuldade na organização das tabelas de jogos dos times participantes. Esse problema é comumente conhecido como Problema de Programação de Jogos (PPJ), e consiste em montar uma tabela de confrontos entre todos os participantes, respeitando uma série de restrições impostas pelos organizadores. O objetivo do presente trabalho é aplicar a meta-heurística híbrida *Clustering Search* (CS) juntamente com a meta-heurística *Simulated Annealing* (SA) de modo a minimizar as distâncias percorridas pelos times durante o torneio visando reduzir os custos associados aos deslocamentos das equipes durante a competição. Para tanto, foram coletados os dados referentes ao Campeonato Brasileiro de Futebol para a modelagem computacional e definição das estruturas para representação de uma solução e, por fim, foram aplicadas as meta-heurísticas *Clustering Search* e *Simulated Annealing* para resolução do Problema de Programação de Jogos referente ao Campeonato Brasileiro de Futebol. A aplicação das meta-heurísticas CS e SA se mostrou eficiente, as soluções produzidas alcançaram valores muito próximos e até melhores que a solução utilizada pela Confederação Brasileira de Futebol (CBF) no ano de 2014, sendo possível atingir o objetivo de reduzir a distância total percorrida pelas equipes ao longo do campeonato. Portanto, a utilização do método proposto é aplicável na prática para tratar o PPJ do Campeonato Brasileiro de Futebol.

Palavras chave: Problema de Programação de Jogos, *Clustering Search,* Meta-heurísticas.

ABSTRACT

Sports confederations leagues have a hard job for organizing games tables of participating teams when programming a competition. This problem is commonly known as Traveling Tournament Problem (TTP), and it consists of setting up a table of confrontations between all participants respecting several constraints imposed by the organizers. The objective of this research consists of applying the hybrid metaheuristic Clustering Search (CS) using the metaheuristic Simulated Annealing (SA) in order to minimize the distances traveled by the teams reducing the costs associated with the journeys of the teams during the competition. Therefore, data from the Brazilian Soccer Championship were collected to perform the computational modeling and definition of the structures needed to represent a solution for the problem. Finally, the metaheuristics Clustering Search and Simulated Annealing were applied to solve the Traveling Tournament Problem related the Brazilian Soccer Championship. The CS and SA were efficient producing solutions close or better than the ones used by the Brazilian Football Confederation (CBF) in the year of 2014, i.e., reducing the total distance traveled by the teams over the championship. Therefore, the use of the proposed method is applicable to solve the TTP related to the Brazilian Soccer Championship.

Keywords: Traveling Tournament Problem, Clustering Search, Metaheuristics.

LISTA DE FIGURAS

Figura 1 - Fluxograma da CS. 22

Figura 2 - Algoritmo SA. 23

Figura 3 - Exemplo de cálculo da distância de Hamming.....25

LISTA DE TABELAS

Tabela 1 - Representação de uma solução envolvendo 6 times. 19

Tabela 2 - Tabela de jogos antes da aplicação do movimento TMC. 20

Tabela 3 - Tabela de jogos resultante da aplicação do movimento TMC. 20

Tabela 4 - Tabela de jogos antes da aplicação do movimento TR. 20

Tabela 5 - Tabela de jogos resultante da aplicação do movimento TR. 21

Tabela 6 - Tabela de jogos antes da aplicação do movimento TO. 21

Tabela 7 - Tabela de jogos resultante da aplicação do movimento TO. 21

Tabela 8 - Penalizações utilizadas na função objetivo. 28

Tabela 9 - Combinação de Parâmetros da CS e SA. 28

Tabela 10 - Melhores parâmetros para a SA e a CS. 29

Tabela 11 - Resultados obtidos após dez execuções da CS. 29

Tabela 12 - Comparação com a solução utilizada pela CBF no ano de 2014. 30

LISTA DE ABREVIATURAS E SIGLAS

CBF – Confederação Brasileira de Futebol

CS – *Clustering Search*

FO – Função Objetivo

ILS – *Iterated Local Search*

MLB – *Major League Baseball*

mTTP – *Mirrored Traveling Tournament Problem*

PPJ – Problema de Programação de Jogos

PPJE – Problema de Programação de Jogos Espelhado

SA – *Simulated Annealing*

TMC – Troca de Mando de Campo

TO – Troca de Oponentes

TR – Troca de Rodadas

TTP – *Traveling Tournament Problem*

SUMÁRIO

1 INTRODUÇÃO 12
1.1 O PROBLEMA E SUA IMPORTÂNCIA 13
1.2 OBJETIVOS 14
1.2.1 **Objetivo geral** 14
1.2.2 **Objetivos específicos** 15
1.3 ESTRUTURA DO TRABALHO 15
2 REVISÃO DE LITERATURA 16
3 METODOLOGIA 18
3.1 COLETA DE DADOS 18
3.2 REPRESENTAÇÃO DE UMA SOLUÇÃO 18
3.3 ESTRUTURAS DE VIZINHANÇA 19
3.3.1 **Troca de Mando de Campo** 19
3.3.2 **Troca de Rodadas** 20
3.3.3 **Troca de Oponentes** 21
3.4 *CLUSTERING SEARCH* 22
3.4.1 **Principais componentes da CS** 23
3.4.1.1 Geradora de soluções 23
3.4.1.2 Processo de agrupamento 24
3.4.1.3 Busca local 25
3.4.2 **Função objetivo** 26
4 RESULTADOS 28
5 CONCLUSÕES 32
REFERÊNCIAS BIBLIOGRÁFICAS 33

1 INTRODUÇÃO

O Problema de Programação de Jogos (PPJ), ou escalonamento de jogos, é muito comum em competições esportivas, tais como basquetebol, futebol, futsal, voleibol, etc. Esse problema consiste em montar uma tabela de jogos entre os participantes de uma competição, respeitando uma série de restrições. Na literatura, esse problema também é conhecido como *Traveling Tournament Problem* (TTP) (DORIGO et al., 2001).

Problemas desse tipo possuem muitas restrições e objetivos a serem alcançados para que a solução seja viável na prática, como a minimização das distâncias percorridas pelos times durante a competição, visando reduzir o custo com os deslocamentos, à realização de apenas uma partida por time e por dia, entre outras. Além disso, é fundamental equilibrar a distância total percorrida pelos times de forma que o seu desempenho não seja afetado devido aos deslocamentos.

Devido ao grande número de restrições, a construção de uma tabela de jogos que seja viável na prática, ou seja, que atenda todas as restrições, é um problema muito difícil de ser resolvido. A dificuldade de solução desse problema é atribuída ao grande número de possibilidades a serem analisadas.

De acordo com Concílio e Zuben (2002), para uma competição envolvendo *n* times, com *n* par, confrontando-se entre si em turnos completos (nos quais todos os times jogam em cada rodada), o número de combinações possíveis é dado por:

$$(n-1)!\,(n-3)!\,(n-5)!\,\ldots\big(n-(n-1)\big)!\times 2^{(n-1)\times\frac{n}{2}}$$

Para exemplificar a magnitude do espaço de soluções, para um torneio com 20 participantes há $2{,}9062\text{x}10^{130}$ combinações possíveis.

Assim, devido ao grande número de possibilidades a serem analisadas, a resolução desses problemas por métodos exatos em tempo computacional razoável é inviável. Porém, esse problema pode ser tratado por métodos heurísticos, que são algoritmos de aproximação e que apesar de não garantirem encontrar a solução ótima para o problema, são capazes de apresentar boas soluções em um tempo viável.

1.1 O PROBLEMA E SUA IMPORTÂNCIA

Inicialmente proposto por Easton et al. (2001), o *Traveling Tournament Problem* é um problema conhecido na literatura como sendo da classe *sports timetabling,* e foi empregado como uma abstração das características da MLB – *Major League Baseball* – dos Estados Unidos. Essa classe de problemas apresenta grandes possibilidades de otimização, uma vez que apresenta uma grande quantidade de restrições que devem ser atendidas para que a solução seja viável na prática, além dos objetivos distintos para se cumprir.

Muitas ligas e confederações esportivas enfrentam esse problema frequentemente ao montarem as tabelas dos jogos dos times associados em cada campeonato. A dificuldade na geração de escalas de jogos para competições cresce quando as sedes das diferentes equipes estão geograficamente distribuídas em uma grande região. Por este motivo, a distância percorrida pelas equipes é uma importante variável a ser minimizada a fim de diminuir as despesas com viagens e dar mais tempo de descanso aos jogadores.

Uma generalização do TTP foi proposta por Ribeiro e Urrutia (2004): o *Mirrored Traveling Tournament Problem* (mTTP), que pode ser traduzido como Problema de Programação de Jogos Espelhado (PPJE). O PPJE apresenta características similares a de torneios de futebol populares na América Latina (como, por exemplo, o Campeonato Brasileiro de Futebol). Em um torneio com turno simples, no qual cada time joga com cada um dos outros exatamente uma vez, são necessárias n – 1 rodadas para completar a escala. Já em torneios com turno e returno, cada time joga com cada um dos outros duas vezes, sendo um jogo em casa e outro fora de casa. O PPJE pode ser considerado um torneio simples em suas n – 1 primeiras rodadas, chamado primeiro turno, seguido pelo mesmo torneio com os mandos de campo invertidos em relação às n – 1 primeiras rodadas, chamado returno, que é o espelho do turno em relação aos mandos de campo. Isso quer dizer que, se um time A jogou em sua sede (ou seja, em sua casa) contra um time B na primeira rodada do turno, ele deverá jogar com o mesmo time B na primeira rodada do returno, porém dentro da sede do time B (neste caso, fora de casa).

O Campeonato Brasileiro de Futebol é um torneio que ocorre anualmente no Brasil e sua tabela de jogos possui o formato de turno e returno, sendo classificado como

PPJE. Uma competição é formada por *n* times, sendo que cada um jogará duas vezes contra os demais, sendo um jogo em sua sede e outro na sede do adversário, não necessariamente nessa ordem. 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 (returno) são os mesmos do primeiro turno, inclusive na mesma ordem, porém com mando de campo invertido (ou seja, um em casa e outro fora de casa, ou vice-versa). Assim, cada time fará $2n - 2$ jogos, sendo um por rodada, totalizando então em $2n - 2$ rodadas ($n - 1$ no turno e $n - 1$ no returno), com $n/2$ jogos cada. Logo, serão realizados $n^2 - n$ jogos. A seguir são apresentadas as restrições para o problema em questão.

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. A ordem dos jogos no turno e returno deverá ser a mesma, porém com mando de campo invertido.
5. Não poderá haver jogos entre times do mesmo estado na última rodada;
6. A diferença entre os jogos feitos em casa e fora de casa de um time, não pode ser maior que uma unidade. Tanto no turno, como no returno;
7. Um time não pode jogar mais que três vezes consecutivas dentro ou fora de casa.

Cada time está associado a um estádio, localizado em uma cidade, considerada sua sede. As distâncias entre as cidades sedes dos times são dadas por uma matriz D, $n \times n$. Cada time inicia o torneio em casa e começa a viajar para cumprir seus jogos nas sedes escolhidas. Cada time retorna, se necessário, para sua sede no fim do torneio.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Este trabalho tem como objetivo principal aplicar a meta-heurística *Clustering Search* para resolução do Problema de Programação de Jogos do Campeonato Brasileiro

de Futebol. Assim, espera-se gerar soluções que atendam às restrições definidas pela Confederação Brasileira de Futebol (CBF), de modo a reduzir a distância total viajada pelos times durante o campeonato visando reduzir o custo financeiro.

1.2.2 Objetivos específicos

Os objetivos específicos são apresentados a seguir.

a. Estudar as regras e restrições do Campeonato Brasileiro de Futebol.

b. Coletar os dados referentes ao campeonato (times, distâncias, etc.).

c. Modelar o problema computacionalmente.

d. Implementar e aplicar a meta-heurística *Clustering Search* ao problema.

f. Avaliar o desempenho do método por meio de experimentos computacionais comparados à tabela utilizada pela CBF no ano de 2014.

1.3 ESTRUTURA DO TRABALHO

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 são apresentadas outras abordagens encontradas na literatura para solucionar o PPJ. O Capítulo 3 define a metodologia utilizada para realização deste trabalho. Os resultados obtidos estão descritos no Capítulo 4 e, por fim, no Capítulo 5 estão as conclusões.

2 REVISÃO DE LITERATURA

O Problema de Programação de Jogos é um problema conhecido na literatura como sendo da classe *sports timetabling*, que abstrai características dos problemas de escalonamento de jogos em competições. Diferentes metodologias são apresentadas na literatura para resolver problemas de programação de jogos.

Problemas relacionados a competições de beisebol são abordados em Cain Jr. (1977) e Russel e Leung (1994); enquanto Nemhauser e Trick (1998), Campbell e Chen (1976), Bean e Birge (1980) e Henz (2001) 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. A seguir, é apresentada uma breve revisão bibliográfica e as abordagens utilizadas em alguns dos principais trabalhos relacionados a esse problema.

Um dos trabalhos precursores que tratam problemas de escalonamento em esportes foi o de Costa (1995). Neste trabalho, o autor 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.

Nemhauser e Trick (1998) trataram o problema da alocação de jogos de um torneio de basquete envolvendo nove times de diferentes localidades dos Estados Unidos utilizando uma combinação de uma técnica de programação inteira e uma técnica enumerativa.

Concílio (2000) abordou o problema da montagem da tabela de jogos do Campeonato Paulista de Futebol de 1997 combinando técnicas de programação genética, computação evolutiva, busca local e técnicas de otimização baseadas em restrições.

Yang et al. (2002) tratam o problema da MLB - *Major League Baseball* - dos Estados Unidos usando uma estratégia evolutiva associada a dois métodos: *Simulated Annealing* (SA) e 2-Opt. O método consiste em aplicar mutações nos indivíduos, e, a cada nova geração, são aplicados o SA e o 2-Opt com o intuito de melhorar os indivíduos gerados.

Anagnostopoulos et al. (2003) também propuseram uma aplicação da meta-heurística *Simulated Annealing* ao PPJ, porém utilizando-a nas instâncias da MLB. O algoritmo obteve resultados significativos, produzindo boas soluções para o problema abordado. O maior problema apresentado pelo algoritmo é o enorme esforço computacional exigido, visto que algumas soluções foram encontradas após dias de processamento.

Biajoli et al. (2003,2004) abordaram com *Simulated Annealing* a primeira divisão do Campeonato Brasileiro de Futebol de 2002 e 2004, respectivamente. O método proposto melhorou significativamente a solução elaborada pela entidade organizadora do evento.

Ribeiro e Urrutia (2004) desenvolveram um método híbrido utilizando as metodologias GRASP (FEO; RESENDE, 1995) e o método *Iterated Local Search* (ILS) (LOURENÇO et al., 2002) para resolver instâncias da literatura relativas ao PPJ espelhado e não-espelhado, bem como uma instância adaptada do Campeonato Brasileiro de Futebol de 2003.

A aplicação da meta-heuristica *Simulated Annealing* também apresentou resultados satisfatórios quando utilizada por Zhang et al. (2004). O método proposto combinou o SA e o algoritmo *Hill-Climbing*, sendo que o SA gerava as escalas e as passava para o *Hill-Climbing*, que por sua vez realizava uma busca local. A melhor solução encontrada era então devolvida para o SA e esse ciclo se repetia até atingir uma condição de parada.

As metodologias apresentadas demonstram a eficiência da aplicação da *Simulated Annealing* em problemas de programação de jogos com diferentes abordagens. Além disso, a aplicação da *Clustering Search* em outros problemas combinatoriais apresentou bons resultados (CHAVES; LORENA, 2010; OLIVEIRA, 2012a,b). Deste modo a aplicação conjunta dessas técnicas ao PPJE é inovadora para o PPJE do Campeonato Brasileiro de Futebol.

3 METODOLOGIA

A metodologia abordada neste trabalho consiste na coleta de dados referentes ao Campeonato Brasileiro de Futebol, na modelagem computacional e definição das estruturas para representação de uma solução e, por fim, o desenvolvimento e aplicação da meta-heurística *Clustering Search* (CS) para resolução do Problema de Programação de Jogos (PPJ) referente ao Campeonato Brasileiro de Futebol.

3.1 COLETA DE DADOS

Inicialmente foram coletados dados referentes à tabela do Campeonato Brasileiro de Futebol do ano de 2014, como: a lista de times participantes, a tabela contendo o escalonamento dos jogos, a sede de cada time de acordo com sua cidade de origem, e as restrições definidas pela Confederação Brasileira de Futebol (CBF).

Após essa fase inicial, foi montada uma tabela contendo as distâncias entre todas as sedes dos times participantes por meio da ferramenta *Google Maps,* essa distância foi dada em quilômetros (Km) e, além disso, a distância entre a cidade sede de um time A para a cidade sede de um time B é diferente da distancia da cidade sede de B para a de A, pelo fato de existirem rotas diferentes tanto no sentido da cidade A para a Cidade B, quando no sentido contrário. A distância entre os times é dada numa matriz D assimétrica n x n, na qual n é o número de times.

3.2 REPRESENTAÇÃO DE UMA SOLUÇÃO

A representação da solução adotada neste trabalho foi proposta por Anagnostopoulos et al. (2003). Seja n o número de times envolvidos, o número de rodadas é dado por $r = 2n - 2$. A solução é representada por uma matriz de n linhas por r colunas, indicando os oponentes de cada time em cada uma das rodadas. Cada linha da tabela 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, ou seja, o sinal positivo indica que o time T_i irá jogar contra um oponente em sua própria sede, e o sinal negativo indica, por sua vez, que o time T_i jogará contra seu oponente na sede do time que está confrontando. Considerando um torneio com 6 times (n = 6 e portanto r = 10,

sendo 5 rodadas no turno e 5 rodadas espelhadas no returno), uma possível escala para este torneio é apresentada na Tabela 1.

Tabela 1 - Representação de uma solução envolvendo 6 times.

	Turno					Returno				
	Rodadas									
Times	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

Pode ser observado na Tabela 1, por exemplo, na primeira rodada do time 1, a notação +6 indica que o time 1 joga em sua sede contra o time 6, enquanto que o time 2, na rodada 2, joga fora de sua sede contra o time 3, representado por -3.

O campeonato de 2014 foi realizado em dois turnos, desse modo todos os times jogaram duas vezes com todos os outros, sendo que um jogo foi realizado no primeiro turno e o outro no segundo turno. Esse campeonato contou com participação de 20 times (n = 20) que se confrontaram em 38 rodadas (r = 38). Uma solução s para o turno deste problema é uma matriz com 20 linhas, 19 colunas e 380 células. Nesse caso, o returno é desconsiderado, pois os jogos deste possuem a mesma disposição dos jogos do turno, porém com o mando de campo invertido.

3.3 ESTRUTURAS DE VIZINHANÇA

As estruturas de vizinhança definem um conjunto de soluções a partir de uma solução s pela aplicação de três movimentos, descritos abaixo. Esses movimentos aplicados à solução s permitem gerar uma solução vizinha s' que será usada pela SA para explorar o espaço de soluções do problema.

3.3.1 Troca de Mando de Campo

O movimento Troca de Mando de Campo (TMC) realiza a troca do local onde será realizada a partida entre dois times t_i e t_j, sendo t_i escolhido aleatoriamente e t_j seu oponente, em uma rodada r_k também escolhida aleatoriamente. O mando de campo é representado pelo sinal positivo ou negativo que acompanha o número que representa o time. Esse movimento é representado pelas Tabelas 2 e 3. Na Tabela

2, o time 1 joga contra o time 6 na rodada 1 na sede do time 1. Já na Tabela 3, o time 1 joga contra o time 6 na rodada 1 na sede do time 6.

Tabela 2 - Tabela de jogos antes da aplicação do movimento TMC.

	Turno					Returno				
	Rodadas									
Times	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

Tabela 3 - Tabela de jogos resultante da aplicação do movimento TMC.

	Turno					Returno				
	Rodadas									
Times	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

3.3.2 Troca de Rodadas

O movimento Troca de Rodadas (TR) realiza a troca entre um rodada r_i e r_j, escolhidas de forma aleatória, permutando todos os times e seus respectivos oponentes entre as duas rodadas. A Tabela 4 representa a tabela de jogos antes da troca da rodada 2 com a rodada 4. A Tabela 5 representa a tabela resultante da aplicação do movimento TR.

Tabela 4 - Tabela de jogos antes da aplicação do movimento TR.

	Turno					Returno				
	Rodadas									
Times	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

Tabela 5 - Tabela de jogos resultante da aplicação do movimento TR.

	Turno					Returno				
	Rodadas									
Times	1	2	3	4	5	6	7	8	9	10
1	+ 6	**+ 3**	+ 4	**- 5**	- 2	- 6	**- 3**	- 4	**+ 5**	+ 2
2	+ 5	**+ 4**	+ 6	**- 3**	+ 1	- 5	**- 4**	- 6	**+ 3**	- 1
3	- 4	**- 1**	+ 5	**+ 2**	+ 6	+ 4	**+ 1**	- 5	**- 2**	- 6
4	+ 3	**- 2**	- 1	**+ 6**	- 5	- 3	**+ 2**	+ 1	**- 6**	+ 5
5	- 2	**- 6**	- 3	**+ 1**	+ 4	+ 2	**+ 6**	+ 3	**- 1**	- 4
6	- 1	**+ 5**	- 2	**- 4**	- 3	+ 1	**- 5**	+ 2	**+ 4**	+ 3

3.3.3 Troca de Oponentes

O movimento Troca de Oponentes (TO) representa a troca do oponente do time t_i com o oponente do time t_j, selecionados aleatoriamente, nas rodadas r_k e r_l, também selecionadas aleatoriamente. A Tabela 7 representa o resultado da troca do oponente do time 1 na rodada 2 com o oponente do time 4 na rodada 4. Em princípio, esse movimento gera soluções inviáveis, porém, essa solução pode vir a se tornar válida com a aplicação posterior deste e de outros movimentos.

Tabela 6 - Tabela de jogos antes da aplicação do movimento TO.

	Turno					Returno				
	Rodadas									
Times	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

Tabela 7 - Tabela de jogos resultante da aplicação do movimento TO.

	Turno					Returno				
	Rodadas									
Times	1	2	3	4	5	6	7	8	9	10
1	+ 6	**- 2**	+ 4	+ 3	- 2	- 6	**+ 2**	- 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	**- 5**	- 5	- 3	- 6	+ 1	**+ 5**	+ 5
5	- 2	+ 1	- 3	- 6	+ 4	+ 2	- 1	+ 3	+ 6	- 4
6	- 1	- 4	- 2	+ 5	- 3	+ 1	+ 4	+ 2	- 5	+ 3

3.4 *CLUSTERING SEARCH*

A *Clustering Search* é um método iterativo que procura dividir o espaço de busca e localizar regiões promissoras por meio do enquadramento dessas em *clusters* (CHAVES; LORENA, 2010).

Um *cluster* pode ser definido por três atributos $C = \{c, v, r\}$. O centro c_i é uma solução que representa o *cluster i*, e identifica a sua localização dentro do espaço de busca. O volume v_i é a quantidade de soluções agrupadas no *cluster i*. Um *cluster* se torna promissor quando o volume atingir um certo limitante λ. O índice de ineficácia r_i é uma variável de controle para identificar se a busca local está ou não melhorando o centro do *cluster i*. O valor de r_i indica o número de vezes consecutivas que a busca local foi aplicada no *cluster i* e não melhorou a solução. Esse atributo evita que a busca local seja executada em regiões ruins ou regiões que já tenham sido suficientemente exploradas por mais de r_{max} vezes. Na Figura 1 está representado o fluxograma de execução da CS com seus principais componentes.

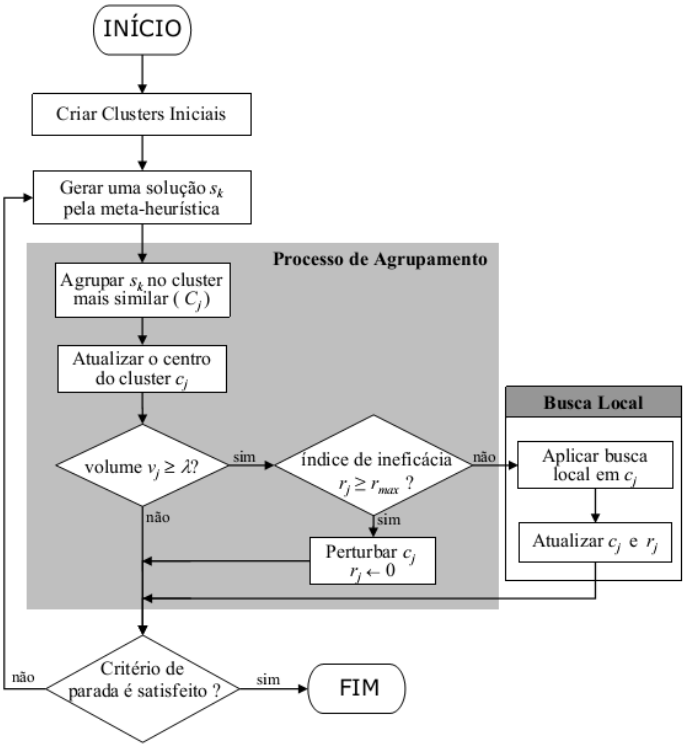

Figura 1 - Fluxograma da CS.
Fonte: Chaves e Lorena (2010).

A CS é formada basicamente por três componentes principais: uma meta-heurística geradora de soluções, um processo de agrupamento e uma heurística de busca

local. A cada iteração da CS, uma solução *s* é gerada pela meta-heurística e enviada para o processo de agrupamento. Essa solução é então agrupada no *cluster* mais similar C_j e o centro desse *cluster* c_i é atualizado, ou seja, a solução antiga é substituída por uma nova fazendo com que ele sofra um distúrbio. Em seguida, é analisado o volume v_i do *cluster* c_i e, caso esse volume atinja um limitante λ ($v_i \geq \lambda$), percebe-se que algum padrão de solução está sendo predominantemente gerado pela meta-heurística. Portanto, esse *cluster* pode estar em uma região de busca promissora. Por fim, é analisado o índice de ineficácia r_j, ou seja, caso a heurística de busca local não melhore a solução por r_{max} vezes consecutivas ($r_j \geq r_{max}$), é aplicada uma perturbação aleatória no centro c_j, objetivando escapar de um possível ótimo local. Por outro lado, se $r_j < r_{max}$, a heurística de busca local é aplicada no centro c_j analisando a vizinhança do *cluster*. Encerrado esse processo, retorna-se para a meta-heurística que irá gerar uma nova solução. O critério de parada da CS é geralmente definido pela meta-heurística escolhida.

3.4.1 Principais componentes da CS

3.4.1.1 Geradora de soluções

A meta-heurística escolhida para fornecer soluções à CS a cada iteração é o *Simulated Annealing* - SA (KIRKPATRICK et al., 1983), apresentado a seguir.

```
1.   Dado (α, SA_max, T_0, T_c e s)
2.   s* ← s            {Melhor solução obtida até então}
3.   iter ← 0 {Número de iterações na temperatura T}
4.   T ← T_0            {Temperatura corrente}
5.   ENQUANTO T > T_C FAÇA
6.      ENQUANTO iter < SA_max FAÇA
7.         iter ← iter + 1          s' ← N(s)       Δ ← f(s') − f(s)
8.         SE Δ < 0 ENTÃO
9.            s ← s'
10.           SE (f(s') < f(s*)) ENTÃO   s* ← s'  FIM-SE;
11.        SENÃO
12.           Com probabilidade e^(−Δ/T) s ← s'
13.        FIM-SE
14.     FIM-ENQUANTO
15.     T ← αT     iter ← 0
16.  FIM-ENQUANTO
17.  Retornar s*
```

Figura 2 - Algoritmo SA.
Fonte: Adaptado de Mauri (2005).

O *Simulated Annealing* é um método de busca local que aceita movimentos de piora como forma de escapar de ótimos locais (KIRKPATRICK et al., 1983). De forma semelhante aos métodos tradicionais de busca local, este método requer que seja definida uma estrutura de vizinhança e uma função de avaliação *f* que associa a cada solução um valor numérico representando o custo da solução. O algoritmo inicia a partir de uma solução inicial gerada de forma aleatória, e seleciona aleatoriamente uma solução *s'* vizinha de *s*. A função objetivo é denotada por *f,* e a cada geração de *s'*, é testada a variação do valor da função objetivo Δ, isto é, $\Delta = f(s') - f(s)$. Se $\Delta < 0$, então $f(s') < f(s)$, logo a solução *s'* apresenta melhor valor numérico com relação à solução corrente *s*, assim o método aceita a solução *s'*, que passa a ser a nova solução corrente *s*. Caso contrário, ou seja, se $\Delta > 0$, é calculada a probabilidade de aceitação $e^{(-\Delta/T)}$, na qual *T* é um parâmetro do método chamado de temperatura e que regula a probabilidade de aceitação de soluções que pioram o valor da função objetivo. O procedimento para quando temperatura chega a um valor próximo de zero e nenhuma solução que seja pior que a melhor solução é mais aceita, isto é, quando o sistema está estável. A solução obtida quando o sistema encontra-se nesta situação evidencia o encontro de um ótimo local.

A solução inicial base para a SA é construída aleatoriamente de modo que sempre respeite três restrições essenciais, são elas: 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.

Essa implementação permite que o método SA receba uma solução com valores baixos de FO e, desse modo, o método tem como parâmetro inicial um solução com baixo custo, permitindo a comparação de soluções com valores melhores do que se o método tivesse recebido uma solução aleatória inválida. Porém, para gerar essa solução inicial válida, é consumido um grande tempo de processamento.

3.4.1.2 Processo de agrupamento

O processo de agrupamento consiste em analisar e alocar soluções ao *cluster* mais similar. Após ser agrupada no *cluster* mais similar C_j, o centro desse *cluster* c_i é

atualizado com informações contidas na nova solução agrupada, fazendo com que o centro se desloque no espaço de busca. A análise de similaridade entre a solução *s* e o centro do *cluster* c_i é calculada de acordo com a distância de Hamming H_i (HAMMING, 1950), em que é utilizado o número de desigualdades encontradas entre uma solução *s* e o centro de um *cluster* c_i. Na Figura 3 é descrito um exemplo do cálculo da distância de Hamming, em que duas soluções são apresentadas. O cálculo é realizado verificando quantas vezes ocorreu alguma diferença entre as soluções.

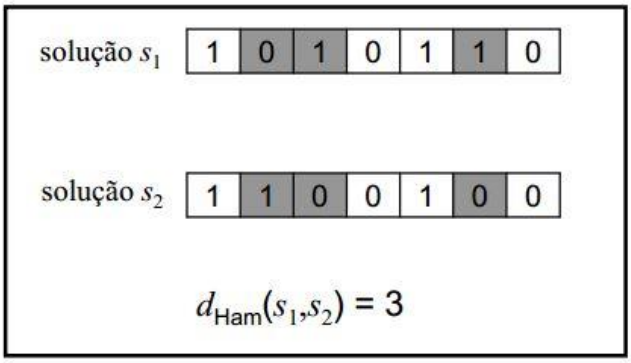

Figura 3 - Exemplo de cálculo da distância de Hamming.
Fonte: Chaves (2009).

Após ser assimilada ao *cluster* com menor número de desigualdades, é verificado se a solução *s* possui um melhor custo em comparação com o centro do *cluster,* caso possua, a solução *s* passa a ser o novo centro do *cluster* c_i.

3.4.1.3 Busca local

Foi implementada uma busca local de melhor melhora (HANSEN; MLADENOVIC, 2006). Essa busca sempre avaliará todas as soluções definidas por uma vizinhança e, ao final da busca, a solução com maior melhora será aceita. A busca consiste na troca dos mandos de campo da solução que representa o centro do *cluster*. Caso essa troca produza uma solução melhor que a atual, essa solução é guardada, e o processo se repete. Ao final da execução, a melhor solução encontrada é reinserida como solução inicial e a busca é reiniciada a partir dela. Caso a troca dos mandos de campo produza uma solução pior que a atual, então esse movimento é desfeito, e a busca continua a partir da solução anterior à troca. A busca local se encerra após realizar a troca de todos os mandos de campo da melhor solução, e não forem

encontradas soluções melhores. Nesse ponto, o centro do *cluster* é atualizado com a melhor solução obtida pela busca local.

3.4.2 Função objetivo

A função objetivo visa minimizar a distância total percorrida pelos times durante uma competição. No presente trabalho, foi utilizada a estratégia de penalização em soluções que violem as restrições impostas ao problema tendo por objetivo evitar que sejam consideradas soluções inviáveis. Assim, a função objetivo é dada por:

$$f(s) = \sum_{i \in T} D_i + \sum_{j \in C} (N_j \times P_j) + dif(d)$$

f(*s*)*:* função que avalia uma solução *s;*

T: conjunto de times participantes da competição;

C: conjunto de restrições que descrevem o problema;

D_i: deslocamento total do time *i;*

N_j: quantidade de vezes que a restrição $J \in C$ não foi atendida;

P_j: penalidade pelo não atendimento à restrição $J \in C$;

dif(*d*)*:* diferença de distância entre o time que mais viajou e o time que menos viajou, dada por:

$$dif(d) = \max\ \{d(i), i\ \in\ T\} - \min\{d(j), j\ \in\ T\}$$

A distância D_i corresponde ao somatório de todas as distâncias de todos os times *i*, em que, a distância de cada time *i* é calculada com base na sua trajetória durante a competição.

Os valores atribuídos às penalidades p_j foram determinados durante a implementação e compreendem o conjunto $P = \{p_1, p_2, p_3, p_4, p_v\}$, sendo que:

p_1: penalidade por violar a restrição 4, de acordo com a Seção 1.1: a ordem dos jogos no turno e returno deverá ser a mesma, porém com mando de campo invertido.

p_2: penalidade por violar a restrição 5: não poderá haver jogos entre times do mesmo estado na última rodada;

p_3: penalidade por violar a restrição 6: a diferença entre os jogos feitos em casa e fora de casa de um time, não pode ser maior que uma unidade, tanto no turno como no returno;

p_4: penalidade por violar a restrição 7: um time não pode jogar mais que três vezes consecutivas dentro ou fora de casa;

p_v: penalidade atribuída a uma solução inválida: uma solução é considerada inválida se desrespeitar alguma das seguintes restrições:

- Cada time joga somente uma vez por rodada (restrição 1);
- Dois times jogarão entre si duas vezes, uma no turno e a outra no returno, alternando o mando de campo entre os mesmos (restrição 2);
- Nas duas primeiras rodadas de cada turno, cada time alternará seus jogos, sendo um em casa e o outro na casa do adversário (restrição 3).

4 RESULTADOS

Os valores atribuídos às penalidades do conjunto *P* utilizadas no calculo da função objetivo foram determinados empiricamente por meio de diferentes testes de modo a produzir soluções que atendam a todo o conjunto de restrições. Dessa forma, quando aplicadas as penalizações com valores muito altos, o algoritmo não aceita soluções que não atendam a todo o conjunto de restrições, ou seja, somente aceita soluções viáveis. Os pesos obtidos estão descritos na Tabela 8:

Tabela 8 - Penalizações utilizadas na função objetivo.

Penalização	Valor
p_1	100.000
p_2	100.000
p_3	100.000
p_4	100.000
p_v	100.000

Os testes para a calibração dos parâmetros das meta-heuristicas *Simulated Annealing* (SA) e *Clustering Search* (CS) foram executados em conjunto com diversas combinações de parâmetros tanto para a CS quanto para a SA, totalizando 486 combinações diferentes. Esses parâmetros foram definidos de forma empírica e estão dispostos na Tabela 9.

Tabela 9 - Combinação de Parâmetros da CS e SA.

SA	SA_{max}	1.000	5.000	10.000
	α	0,975	0,995	-
	T_0	100	1.000	10.000
	T_C	0,001	-	-
CS	γ	5	7	10
	λ	3	5	7
	r_{max}	3	5	7

Os parâmetros da SA são: $SA_{\max}$ indica a quantidade de iterações da SA (1.000, 5.000 ou 10.000), α é a taxa de resfriamento (0,975 ou 0,995), T_0 é a temperatura inicial (100, 1.000 ou 10.000) e T_c é a temperatura de congelamento (0,001). Já os parâmetros da CS são: γ indica a quantidade máxima de *clusters* (5, 7 ou 10), λ o volume máximo de soluções em cada *cluster* (3, 5 ou 7) e r_{max} o limite para o índice

de ineficácia (3, 5 ou 7). Para cada uma dessas combinações, foram executados 3 testes sem restrição de tempo. Nesse caso, quando o algoritmo finaliza a sua execução, o tempo decorrido para cada execução é computado. Ao final de todas as execuções, foi possível obter os parâmetros para a CS e SA de acordo com a melhor solução gerada entre todas as combinações, conforme Tabela 10.

Tabela 10 - Melhores parâmetros para a SA e a CS.

SA	SA_{max}	5.000
	α	0,975
	T_0	1.000
	T_C	0,001
CS	γ	7
	λ	3
	r_{max}	3

A partir desta combinação de parâmetros da Tabela 10, foram realizados novos testes, sendo a CS executada 10 vezes, sem restrição de tempo. A Tabela 11 apresenta de forma detalhada os resultados obtidos nesta etapa de testes, sendo que a melhor solução entre as 10 execuções está destacada em negrito.

Tabela 11 - Resultados obtidos após dez execuções da CS.

Execução	**FO**	**Tempo (seg)**	**Dist. Total (Km)**	**Dist. do time que mais viajou (Km)**	**Dist. do time que menos viajou (Km)**	**Qtd. de restrições violadas**
1	**716.546**	**159,0**	**680.350**	**60.423**	**24.227**	**0**
2	770.695	209,0	729.483	68.382	27.170	0
3	748.315	191,0	707.258	67.951	26.894	0
4	740.263	186,0	693.771	71.130	24.638	0
5	743.386	185,0	706.785	63.183	26.582	0
6	742.955	178,0	702.422	65.098	24.565	0
7	752.233	293,0	716.263	63.160	27.190	0
8	747.213	193,0	704.151	67.144	24.082	0
9	747.008	221,0	705.338	64.864	23.194	0
10	763.552	203,0	725.135	63.238	24.821	0

Na Tabela 11, a coluna *Execução* identifica cada execução, a coluna *FO* indica o valor da função objetivo de cada solução, a coluna *Tempo* indica o tempo em

segundos que a execução levou para ser finalizada, a coluna *Dist. Total* indica a soma das distâncias percorridas por todos os times durante o campeonato, a coluna *Dist. do time que mais viajou* indica a distância total percorrida pelo time que mais viajou em relação aos outros, a coluna *Dist. do time que menos viajou* indica a distância total percorrida pelo time que menos viajou em relação aos outros, a coluna *Qtd. de restrições violadas* indica o número de restrições não atendidas.

De acordo com os resultados obtidos da Tabela 11, é possível observar que a escolha dos altos valores para as penalizações impediram que a CS aceitasse soluções que violasse alguma restrição, sendo construídas somente soluções viáveis e que não violam nenhuma das restrições impostas, gerando assim soluções de boa qualidade. Outro importante resultado foi o baixo tempo computacional em todas as execuções.

Observando a coluna *Dist. do time que mais viajou* percebe-se que houve pouca variação entre as execuções, o mesmo ocorre com a coluna *Dist. do time que menos viajou*. Uma maior variação ocorre quando comparamos essas duas colunas, esse resultado demonstra a dificuldade em solucionar o problema de equilibrar as distâncias percorridas pelos times de modo a não prejudicar o desempenho dos competidores pelo desgaste com as viagens.

A Tabela 12 apresenta um comparativo entre a melhor solução obtida nas 10 execuções e a solução usada no Campeonato Brasileiro de Futebol do ano de 2014.

Tabela 12 - Comparação com a solução utilizada pela CBF no ano de 2014.

Execução	**FO**	**Tempo (seg)**	**Dist. Total (Km)**	**Dist. do time que mais viajou (Km)**	**Dist. do time que menos viajou (Km)**	**Qtd. de restrições violadas**
Melhor FO	**716.546**	**159,0**	**680.350**	**60.423**	**24.227**	**0**
CBF 2014	721.093	-	690.984	57.341	27.232	0

Como pode ser observado nas Tabelas 11 e 12, os resultados obtidos pela CS possuem valores próximos ao valor apresentado pela solução utilizada pela CBF no ano de 2014. Além disso, o melhor resultado obtido dentre as dez execuções superou a solução utilizada pela CBF, com uma variação de 4.547 no valor da FO e

uma variação de 10.634 Km entre as distâncias totais das duas soluções. Porém, é importante destacar que a solução utilizada pela CBF apresenta melhor balanceamento entre as distâncias percorridas pelo time que mais viajou e o que menos viajou.

5 CONCLUSÕES

Neste trabalho foi tratado o Problema de Programação de Jogos (PPJ) do Campeonato Brasileiro de Futebol em sua versão espelhada conhecida como Problema de Programação de Jogos Espelhado (PPJE). Foram consideradas as restrições definidas pela Confederação Brasileira de Futebol (CBF), sendo o principal objetivo minimizar a distância total viajada pelos times durante o campeonato.

Para a solução do PPJ foi utilizada a combinação das meta-heuristicas *Clustering Search* (CS) tendo como sua geradora de soluções a *Simulated Annealing* (SA).

Os resultados obtidos mostraram que a aplicação das meta-heuristicas CS e SA para solucionar o problema de programação de jogos foi eficiente. Os altos valores das penalizações permitiram que a CS produzisse somente soluções viáveis que atendessem a todas as restrições impostas ao problema do Campeonato Brasileiro de Futebol. Dessa forma, foi possível obter resultados muito próximos e até melhores que os da tabela oficial utilizada pela CBF no ano de 2014. A melhor solução obtida pela CS atingiu uma economia de 10.634 Km quando comparada à solução usada pela CBF, o que na prática representa uma redução direta nos custos do campeonato. Além disso, soluções foram geradas em um tempo computacional muito baixo, tornando sua aplicação prática viável.

Para trabalhos futuros, recomenda-se o uso da CS combinada com outras meta-heurísticas geradoras de soluções.

REFERÊNCIAS BIBLIOGRÁFICAS

ANAGNOSTOPOULOS, A.; MICHEL, L.; HENTENRYCK, P. V.; VERGADOS, Y. A simulated annealing approach to the traveling tournament problem. In: **Proceedings**. [S.l.]: Proceedings Of Cpaior'03, 2003.

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

BIAJOLI, F. L.; MINE, O. M.; CHAVES, A. A.; SOUZA, M. J. F. Escala de jogos de torneios esportivos: uma abordagem via simulated annealing. In: **Simpósio Brasileiro de Pesquisa Operacional**, 35, 2003, Natal. Anais... SBPO, 2003.

BIAJOLI, F. L.; MINE, O. M.; CHAVES, A. A.; SOUZA, M. J. F.; LUCENA, A.; CABRAL, L. A. F.; PONTES, R. C. V. Scheduling the brazilian soccer championship: a simulated annealing approach. In. **Practice and Theory of Automated Timetabling**, 5, 2004, Pittsburgh. Proceedings… PATAT, 2004.

CAIN Jr, W.O. A Computer Assisted Heuristic Approach Used to Schedule the Major League Baseball Clubs. In: **Optimal Strategies in Sports**, p. 32-41, 1977.

CAMPBELL, R. T.; CHEN, D. S. A Minimum Distances Basketball Scheduling Problem. In: **Management Science in Sports**, p. 15-25, 1976.

CHAVES, A. A.; LORENA, L. A. N. Clustering search algorithm for the capacitated centered clustering problem. In: **Computers & Operations Research**, v. 37, p. 552-558, 2010.

CHAVES, A. A. Uma meta-heurística híbrida com busca por agrupamentos aplicada a problemas de otimização combinatória. **Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos**, 2009.

CONCÍLIO, R. Contribuições À Solução De Problemas De Escalonamento Pela Aplicação Conjunta De Computação Evolutiva E Otimização Com Restrições. **Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação - UNICAMP**, p. 37-38, 2000.

CONCÍLIO, R.; ZUBEN, F. J. Uma abordagem evolutiva para geração automática de turnos completos em torneios. In: **Revista Controle & Automação**, v. 13, n. 2, p. 105-122, 2002.

COSTA, D. An evolutionary tabu search algorithm and the NHL scheduling problem. In: **INFOR**, v. 33, n. 3, p. 161-178, 1995.

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

DORIGO, M.; NEMHAUSER, G. L.; TRICK, M. A. The traveling tournament problem: description and benchmarks. **Lecture Notes in Computer Science**, v. 2239, p. 580-585, 2001.

EASTON, K.; NEMHAUSER, G.; TRICK, M. The traveling tournament problem: Description and benchmarks. In: **Seventh International Conference On The Principles And Practice Of Constraint Programming** (CP'99), 2001.

HAMMING, R. W. Error detecting and error correcting codes. In: **Bell System Technical Journal**, v. 26, p. 147-160, 1950.

HANSEN, P.; MLADENOVIC, N. First vs. best improvement: an empirical study. In: **Discrete Applied Mathematics**, v. 154, p. 802-817, 2006.

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

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

MAURI, G. R. Novas heurísticas para o problema de escalonamento de tripulações. Dissertação (Mestrado em Computação Aplicada) – **Instituto Nacional de Pesquisa Espacial, São José dos Campos**, 2005.

NEMHAUSER, G. L.; TRICK, M. A. Scheduling a major college basketball conference. In: **Operations Research**, v. 46, n. 1, p. 1-8, 1998.

OLIVEIRA, R. M.; MAURI, G. R.; LORENA, L. A. N. Clustering search for the berth allocation problem. In: **Expert Systems with Applications**, v. 39, n. 5, p. 5499-5505, 2012a.

_____. Clustering search heuristic for solving a continuous berth allocation problem. In: **Lecture Notes in Computer Science**, v. 7245, p. 49-62, 2012b.

RIBEIRO, C.; URRUTIA, S. Heuristics for the mirrored traveling tournament problem. In. **Practice and Theory of Automated Timetabling**, 5, 2004, Pittsburgh. Proceedings… PATAT, 2004.

RUSSELL, R. A.; LEUNG, J. M. Devising a cost effective schedule for a baseball league. In: **Operations Research**, v. 42, p. 614-625, 1994.

SCHAEFER, A. Scheduling Sport Tournaments using Constraint Logic Programming. In: **Constraints**, v. 4, p. 43-65, 1999.

SCHREUDER, J. A. M. Combinatorial Aspects of Construction of Competition Dutch Professional Football Leagues. In: **Discrete Applied Mathematics**, v. 35, p. 301-312, 1992.

YANG, J. T.; HUANG, H.; HORNG, J. T. Devising a cost effective baseball scheduling by evolutionary algorithms. In: **Proceedings of the 2002 Congress on Evolutionary algorithms**. p. 1660–1665, 2002.

ZHANG, H.; LIM, D.; SHEN, H. A Simulated Annealing and Hill-Climbing Algorithm for the Traveling Tournament Problem. In: **Departmet of IEEM**, p. 40, 2004.