A description...

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL REI

UFSJ

**Problema de otimização para a tabela do Campeonato Brasileiro de 2013**

Bruno Bernardes da Costa

Edson Coelho do Santos

SÃO JOÃO DEL REI, MG  
Julho/2013

**Sumário:**

1.Introdução........................................................................................................

2. Proposta do trabalho.........................................................................................................................

3. Descrição da solução........................

4. Análise e discussões............................

5. Conclusão..........................................

6. Bibliografia....................................

**1. Introdução:**

O campeonato brasileiro de futebol é um torneio do tipo de dois turnos espelhados, assim todos os times jogam contra todos os outros times no primeiro turno em uma localidade(casa ou fora) e no segundo turno jogam os mesmos mesmo jogos na mesma ordem, porém a localidade é invertida, ou seja, o time que jogou em casa na primeira rodada do primeiro turno jogará na primeira rodada do segundo turno fora de casa.

O problema para montar a tabela do Campeonato Brasileiro, onde a distância total percorrida por todos os times é a menor possível é determinado como um problema de otimização, já que para garantir a melhor solução é necessário testar todos os caminhos para todos os times. Dessa forma um problema de otimização tenta calcular por meio de uma heurística uma distância total próxima da solução ótima, porém sem ter que testar todos os caminhos possíveis.

Uma das heurísticas utilizadas para resolver um problema de otimização é o algoritmo GRASP(Greedy Randomized Adaptive Search Procedure). A aplicação do GRASP consiste em criar uma solução inicial e depois efetuar uma busca local para melhorar a qualidade da solução. A solução para um problema de otimização no GRASP é dada na forma de uma conjunto com elementos que atendam todas as restrições existentes no problema. Começa-se com um conjunto vazio, e são inseridos elementos nesse conjunto até que ele represente uma solução viável para o problema. Para cada iteração, todos os elementos candidatos são avaliados segundo uma função gulosa que meça o benefício da inserção desse elemento para a construção da solução. Assim a solução é montada contendo os elementos com melhor valor na função gulosa. A solução final será o melhor elemento definido pela função gulosa.

**2. Proposta do trabalho:**

Para esse trabalho prático nos foi proposto desenvolver duas heurísticas para resolver o problema de viagem do torneio espelhado do campeonato brasileiro de futebol. O campeonato brasileiro de futebol é definido da seguinte forma: dados n times e as distâncias entre as suas cidades de origem, o problema consiste em definir uma sequência, onde um time joga com cada um dos outros duas vezes, uma em casa e outra fora de casa, de tal forma que nenhum time jogue mais de dois jogos fora de casa, além disso não há repetições (ou seja, dois jogos consecutivos entre os mesmos times em localidades diferentes), não devem ser realizados nas últimas três rodadas jogos clássicos regionais. A última restrição para o problema é: os jogos da rodada k são os mesmos da rodada k + (n − 1), k = 1, . . . , n − 1, com a localização invertida.

Dessa forma devemos por meio de uma entrada com os dados das distâncias entre as cidades para montar uma tabela para o Brasileiro 2013, que imprima as rodadas, a distância percorrida por cada time e a distância total percorrida.

**3. Descrição da solução:**

**3.1 Associação dos nomes dos algoritmos:**

* Função main: tp2.c
* Função GRASP: grasp.c
* Função Triangulação de jogos: triangulos.c

**3.2 Implementações:**

O algoritmo tem formato de execução da forma:

./tp2 -i arquivo\_distancias.txt heurística

Assim o algoritmo é iniciado alocando uma matriz dinamicamente de dimensões 20x20, está matriz é usada para ler o arquivo de entrada com as distâncias entre todos os times do campeonato brasileiro de 2013. A matriz com todas as distâncias é espelhada para facilitar os cálculos. De acordo com o valor da heurística no comando de execução, o algoritmo escolhe de qual forma o problema será solucionado, para o valor de heurística 1 o problema será solucionado por triangulação de jogos e o valor 2 o problema é solucionado pelo GRASP.

**3.3 GRASP:**

O algoritmo GRASP utiliza uma matriz joga[20][19], nessa matriz são definidos os locais onde os jogos vão ocorrer(casa ou fora), uma matriz rodada[20][20], onde é definido a tabela com todos os jogos de um turno do campeonato e qual rodada ocorrerá cada jogo, um vetor distCalculadas[20], que armazena a distância total percorrida em um turno para cada time e um outro vetor vp[20], que é o vetor de probabilidade para montar a tabela de jogos. Além dessas variáveis são usadas q0, que é a condição de parada do backtracking implementada na função local, a variável total, que contém a distância total percorrida por todos os times.

A implementação consite em montar a tabela de rodadas e de localidades para 100 tipos diferentes de combinações do vetor de probabilidades, dessa forma a menor distância total percorrida é obtida e a tabela do campeonato brasileiro de 2013 é montada.

O pseudo-código para o algoritmo do GRASP segue abaixo:

repete 100 vezes{

randoniza o vetor de probabilidades;

monta a tabela de rodadas;

monta a tabela de localidades;

total = a distância total percorrida por todos os times

se (total<total\_min) total\_min = total

}

**3.3.1 Descrições das funções:**

* ordenavetor(v);

A função ordena vetor é usada para randomizar o vetor de probabilidades. Ela utiliza a função rand() com a semente dada por srand(time(Null)), assim para cada execução o valor da semente será diferente.

* poligonos(vp,rodadas);

A função polígonos monta a tabela de rodadas para todos os times, retornando a tabela montada na matriz rodada. Os times são distribuídos nos vértices do polígono e um time fica de fora, depois de montado este esquema cada rodada é um estado do polígono, o time que ficou de fora é mantido fixo e os times que formam o polígono são girados.

O seguinte esquema explica melhor o método dos polígonos:

Desenho

* local(rodada, joga, rodada\_inicial, condição\_parada);

A função local monta a tabela de localidade de cada jogo, retornando a tabela na matriz joga. Para montar a matriz joga foi utilizada a técnica de programação backtracking, dessa forma a matriz é montada de forma a não ter nenhum jogo que joga duas vezes fora e duas vezes em casa. A função é encerrada quando toda a tabela de locais é preenchida de forma correta.

* calculoDistancias(rodada,joga,distCalculadas);

A função calculoDistancias calcula as distâncias totais para cada time e a distância total percorrida por todos os times no campeonato. A distância é calculada multiplicando o local de cada jogo contido na matriz joga pela distância de um time até o outro na matriz distancias. O resultado dessa função é usado também para saber qual o menor valor da distância total percorrida por todos os times.

**3.4 Triangulação de jogos:**

Para a função de triangulação de jogos foram usadas as matrizes joga[20][19], nessa matriz são definidos os locais onde os jogos vão ocorrer(casa ou fora), a matriz joga2[20][20], é a transformação da matriz joga para o calculo das distâncias, a matriz rodada[20][20], onde é definido a tabela com todos os jogos de um turno do campeonato e qual rodada ocorrerá cada jogo, a matriz dist[20][20], , e o vetor de probabilidades vp[20].

O pseudo-código descrevendo a triangulação de jogos segue abaixo:

void triangulos(){

randoniza o vetor de probabilidades;

monta a tabela de rodadas;

enquanto(a matriz rodada não for válida){

corrige o time com as três rodadas inválidas;

monta a tabela de rodadas corrigida;

}

monta a tabela de localidades;

mede a distância percorrida para o primeiro turno;

mede a distância percorrida para o segundo turno;

}

**3.4.1 Descrição das funções:**

* ordenavetor(v);

A função ordena vetor é usada para randomizar o vetor de probabilidades. Ela utiliza a função rand() com a semente dada por srand(time(Null)), assim para cada execução o valor da semente será diferente.

* poligonos(vp,rodadas);

A função polígonos monta a tabela de rodadas para todos os times, retornando a tabela montada na matriz rodada. Os times são distribuídos nos vértices do polígono e um time fica de fora, depois de montado este esquema cada rodada é um estado do polígono, o time que ficou de fora é mantido fixo e os times que formam o polígono são girados.

O seguinte esquema explica melhor o método dos polígonos:

Desenho

* local(rodada, joga, rodada\_inicial, condição\_parada);

A função local monta a tabela de localidade de cada jogo, retornando a tabela na matriz joga. Para montar a matriz joga foi utilizada a técnica de programação backtracking, dessa forma a matriz é montada de forma a não ter nenhum jogo que joga duas vezes fora e duas vezes em casa. A função é encerrada quando toda a tabela de locais é preenchida de forma correta.

* medirD(rodada,joga2,dist);

A função medirD mede as distâncias avaliando a cidade de partida do time e o último jogo de forma a obter menos viagens, ou seja, se o time jogou o jogo anterior fora de casa e o próximo jogo for fora também este time parte da cidade em que ele jogou para a próxima sem voltar em casa, se o time jogou em casa o jogo anterior e o próximo jogo for fora a distância é calculada normalmente.

* ordenatriangulo(vp);

A função ordenatriangulo ordena o vetor de probabilidades de forma que o primeiro elemento do vetor armazene o time que mais tem times com distâncias 0 dele. Para os outros elementos do vetor são armazenados em ordem crescente dos times com as menores distâncias para os outros times de forma que o último time seja o mais isolado. Para calcular o time que mais tem times a distâncias 0 é usada a função maisrepete(); e para calcular os outros elementos é usado a função getlesserD(linha, vp).

**4. Análise e discussões:**

**4.1 Complexidade das funções:**

O cálculo das complexidades será separado por função já que várias funções são reutilizadas para as duas Heurísticas.

* ordenavetor(v);

A função ordenavetor depende da quantidade de times do campeonato, é feito um loop para randonizar os times no vetor vp. Logo a função é O(n).

* poligonos(vp,rodadas);

A função poligonos monta a tabela de rodadas para todos os jogos, para isso ele faz um loop

* local(rodada, joga, rodada\_inicial, condição\_parada);
* medirD(rodada,joga2,dist);
* ordenatriangulo(vp);

**5. Conclusão:**

**6. Bibliografia:**

* ZIVIANI, N. . Projeto de Algoritmos com Implementações em Pascal e C. 2a..

ed. São Paulo: Thompson Learning, 2004.