Algorítimos e Estrutura de Dados III
O Problema da Viagem do Torneio

Pablo Cecilio Oliveira
Alexander Cristian

1 Introdução

Otimização é o processo de melhorar algo. Basicamente é a ferramenta de matemática que nós confiamos para obter respostas aos problemas. Em matemática e ciência da computação, um problema de otimização é o problema de encontrar a melhor solução de todas as soluções viáveis. A terminologia "melhor" solução implica que há mais de uma solução e as elas não são de igual valor. Esses problemas de otimização no domínio científico e da vida real encontram soluções em pesquisa algorítmica que mostra o uso de métodos clássicos, métodos heurísticos e baseados na natureza métodos para calcular a solução.

Na solução de problemas difíceis onde não há soluções exatas eficientes, abordagens heurísticas como Simulated Annealing, Busca Tabu, GRASP (Greedy Randomized Adaptive Search Procedure), VNS (Variable Neighborhood Search), VND (Variable Neighborhood Descent), o ILS (Iterated Local Search) e o Método de Reconexão por Caminhos (Path Relinking) se tornam alternativas populares.

Então neste trabalho nós apresentamos um algoritmo que faz uso de uma dessas abordagens, para resolver um problema de otimização conhecido como Traveling Tournament Problem.

1.1 Especificação do problema

O Problema da viagem do torneio (Traveling Tournament Problem) é um problema de agendamento de esportes que abstrai as questões importantes na criação horários, onde a viagem em equipe é uma questão importante. Ligas esportivas profissionais existem em todo o mundo. Essas ligas são muitas vezes de grande importância econômica devido às enormes receitas geradas pela venda de bilhetes e direitos de transmissão para os jogos. Portanto, o planejamento dessas ligas é de suma importância. Um aspecto importante é a geração de um calendário para os torneios que especifica a ordem em que as equipes jogam entre si durante a temporada e o local de cada jogo.

Portanto consideramos um torneio de n times, onde n é um número par. Em um torneio simple round-robin (SRR), cada time joga com cada um dos demais exatamente

uma vez, nas n - 1 rodadas pré-estabelecidas. O jogo entre os times i e j é representado pelo par não ordenado i, j. Há n=2 jogos em cada rodada. Cada time joga exatamente uma vez por rodada. Em um torneio double round-robin (DRR), cada time joga com cada um dos outros duas vezes, uma em casa e outra fora de casa. Um torneio mirrored double round-robin (MDRR) é um torneio SRR nas primeiras n – 1 rodadas, seguido das mesmas rodadas iniciais mas invertendo a sua localização. Uma viagem é uma sequência de jogos fora de casa. Enquanto uma estadia é uma sequência de jogos em casa. Assumimos que cada time tem um estádio para seu uso na sua cidade natal. As distâncias entre as cidades são conhecidas. Cada time parte de sua cidade no início do torneio e retorna à sua cidade no fim do torneio, se já não estiver lá. Sempre que duas rodadas de um time forem fora de casa, ele não retorna à sua cidade natal entre elas.

O problema da viagem do torneio (Traveling Tournament Problem) é definido como a seguir. Dados n times e as distâncias entre as suas cidades de origem, o problema consiste em definir uma sequência DRR de tal forma que nenhum time joga mais de três jogos fora de casa, não há repetições (ou seja, dois jogos consecutivos entre os mesmos times em localidades diferentes), e a soma das distâncias percorridas pelos times é minimizada. O problema da viagem do torneio espelhada tem uma restrição adicional: os jogos da rodada k são os mesmos da rodada k + (n- 1), k = 1; . . . ; n - 1, com a localização invertida.

1.2 Solução proposta

Para a resolução deste problema utilizamos o método de GRASP (Greedy Randomized Adaptive Search Procedures) que é uma meta-heurística constituída por heurísticas construtivas e busca local. Consiste de múltiplas aplicações de busca local, cada uma iniciando de uma solução diferente. As soluções iniciais são geradas por algum tipo de construção randômica gulosa ou algum esquema de perturbação, esse método é constituído basicamente de duas fases: uma fase de construção e uma fase de busca local, cujo objetivo é convergir à solução encontrada na fase de construção para um ótimo local.

Na segunda fase ocorre o refinamento da solução gerada pela fase de construção, aplicando um método de busca local. Quanto melhor é a qualidade da solução gerada pela heurística de construção, maior é a velocidade de convergência desta solução para um ótimo local.

2 Implementação

Inicialmente os dados contidos no arquivo de entrada são verificados. Então é retornado o número de linhas do arquivo de clubes e por sequência o número de clubes. São carregados para um struct array os nomes das sedes e das cidades, e as distancias das cidades são carregadas para um 2Darray

Logo em seguida é utilizado o método do polígono parar gerar uma tabela valida,

que consiste em inicialmente, os n-1 times abstratos serem colocados consecutivamente em sentido horário nos nós numerados de um polígono regular com n-1 nós. Os times 1,2,...,n-1 são colocados nos nós 1,2,...,n-1 respectivamente, enquanto que o time n não é inserido no polígono. A cada rodada k = 1,2,...,n-1 o time localizado no nó l = 2,...,n/2 joga contra o time localizado no nó n + 1 – l. Para toda rodada, o time localizado no nó 1 joga contra o time n. Depois de cada rodada, cada time 1,2,...,n-1 é movido em sentido horário para o próximo nó do polígono.

Posteriormente, a escala é duplicada. Essa escala é utilizada para gerar uma matriz quadrada n x n de oponentes consecutivos. Cada célula (i, j) da matriz contém o número de vezes em que os times i e j são oponentes consecutivos dos outros times.

Aloca-se dinamicamente essa matriz. Depois é retornada a distância de uma sede para outra de determinado clube e então a função “custos” avalia os custos de deslocamentos de cada time. Com a tabela formada associa-se os times com mais jogos consecutivos aos clubes com sedes mais próximas.

A função “count” retorna o número de tentativas para gerar uma tabela com mandos validos. Então gera-se o mando de campo a partir de uma primeira rodada com mando aleatório, e são definidos os custos da solução inicial.

Logo depois aplica-se a solução gerada pela fatoração a um número de interações com o mando de campo aleatório a fim de encontrar uma solução melhor

3 Análise de complexidade

3.1 Funções da Lista Encadeada Simples

Tabela 1: Funções da Lista Encadeada Simples.

Função	Finalidade	Complexidade
create()	Inicializa uma Lista Encadeada Simples	$O(1)$
atP()	Dada a posição, retorna um nó da Lista.	$O(n)$
LLmin()	Retorna um nó com o menor valor de um dado.	$O(n)$
LLmax()	Retorna um nó com o maior valor de um dado.	$O(n)$
LLidx()	Retorna a posição de um nó.	$O(n)$
LLpsh()	Insere os dados a partir da posição cabeça.	$O(1)$
LLpop()	Remove os dados a partir da posição cabeça.	$O(1)$
LLdel()	Remove um nó, dado sua posição na Lista.	$O(n)$
LLins()	Insere um nó, dado sua posição na Lista.	$O(n)$
LLchg()	Troca a posição de dois nós.	$O(n)$
LLdup()	Remove os nós com dados duplicados.	$O(n)$
LLinc()	Ordena a Lista em ordem crescente.	$O(n)$
LLdec()	Ordena a Lista em ordem decrescente.	$O(n)$
LLprt()	Retorna a impressão da Lista.	$O(n)$
LLclr()	Libera os dados da Lista da memoria.	$O(n)$
isEmpty()	Retorna se uma Lista está ou não vazia.	$O(1)$

Fonte: autores

3.2 Funções de fatoração inicial

3.3 Funções de associação dos clubes

3.4 Funções de mando de campo

3.5 Funções de uso comum ao programa

4 Análise de resultados

5 Considerações finais

Durante o desenvolvimento vários métodos foram testados com o objetivo de encontrar a maneira mais eficiente de solucionar o problema da viagem do torneio. //comentários sobre as dificuldades no desenvolvimento do código

Referências

[1] Marcio Tadayuki Mine. Programação de jogos de competições esportivas. http://www.decom.ufop.br/prof/marcone/projects/ttp/Publications/RT_FAPEMIG_Mine_2006.pdf. [Acesso em: 07-Outubro-2018].

O histórico do desenvolvimento desse trabalho se encontra online em: https://github.com/Durfan/ufsj-aeds3-tp2.