MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE

NOVAS HEURÍSTICAS PARA O PROBLEMA DE GERAÇÃO DE ESCALAS DE JOGOS PARA TORNEIOS ESPORTIVOS

Fabrício Lacerda Biajoli

Dissertação de Mestrado em Computação Aplicada, orientada pelo Dr. Luiz Antonio Nogueira Lorena.

INPE
São José dos Campos
2007

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6911/6923
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE

NOVAS HEURÍSTICAS PARA O PROBLEMA DE GERAÇÃO DE ESCALAS DE JOGOS PARA TORNEIOS ESPORTIVOS

Fabrício Lacerda Biajoli

Dissertação de Mestrado em Computação Aplicada, orientada pelo Dr. Luiz Antonio Nogueira Lorena.

INPE
São José dos Campos
2007

00.000.00(000.0)

Instituto Nacional de Pesquisas Espaciais (INPE). Serviço de Formação e Documentação (SID).

NOVAS HEURÍSTICAS PARA O PROBLEMA DE GERAÇÃO DE ESCALAS DE JOGOS PARA TORNEIOS ESPORTIVOS/ Instituto Nacional de Pesquisas Espaciais (INPE). Serviço de Formação e Documentação (SID). – São José dos Campos: INPE, 2007.

107p.; (INPE)

1. *Traveling Tournament Problem* . 2. *Mirrored Traveling Tournament Problem*. 3. Geração de Escalas de Jogos para Torneios Esportivos. 4. *Evolutionary Clustering Search* 5. *Clustering Search* 6. Algoritmos Evolutivos. 7. Heurísticas.

Aprovada pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de Mestre em Computação Aplicada.

Dr. José Demísio Simões da Silva _____

Presidente
INPE

Dr. Luiz Antonio Nogueira Lorena _____

Orientador
INPE

Dr. Luiz Ricardo Pinto _____

Convidado
UFMG

Dr. Geraldo Ribeiro Filho _____

Convidado
FBES

"Cada problema que resolvi, tornou-se uma regra, que serviu depois para resolver outros problemas."

René Descartes

Dedico este trabalho aos meus pais,
Manoel Biajoli e Onícia Lacerda Biajoli,
pois esta conquista é deles também.

AGRADECIMENTOS

Em primeiro lugar, a Deus, pela oportunidade de chegar até aqui.

À minha família, por todo apoio e incentivo durante a realização do mestrado em Computação Aplicada. Em especial, a meus pais, por sempre acreditarem em mim.

Ao Prof. Dr. Luiz Antonio Nogueira Lorena, pela orientação e confiança na minha capacidade para a realização deste trabalho.

Ao Instituto Nacional de Pesquisas Espaciais - INPE, pela chance de aprofundar meus conhecimentos.

A todos os professores do LAC/INPE, pelos ensinamentos transmitidos.

Aos membros da banca examinadora pela disposição em analisar este trabalho.

Ao Prof. Dr. Marcone Jamilson Freitas Souza (UFOP), pela formação e motivação à realização do curso de mestrado.

Ao Prof. Dr. Luiz Ricardo Pinto (UFMG), pela oportunidade de realizar um trabalho de Iniciação Científica durante a graduação, o qual me despertou o interesse pela área de Pesquisa Operacional.

À Mariana, pelo carinho, amor, compreensão e apoio durante estes anos de muito esforço e estudo.

Aos amigos Kçapa, Sidão, Brieba, Kabron, Gilberto e Bruno, pela receptividade e amizade. Ao amigo e irmão Piuí, pela convivência e amizade durante estes anos de estudo.

Aos amigos Inpeanos, pelo convívio, momentos de estudo e descontração e, principalmente, pela troca de experiências.

Aos amigos da Prima Informática, pela amizade e conselhos compartilhados; e ao Eduardo e Walter pela confiança e oportunidade de conciliar trabalho e mestrado.

À República K-Zona e a todos os Zoneiros, pelo aprendizado e convivência constante, que certamente me ajudaram a chegar até aqui.

A todos que de alguma forma contribuíram para a realização deste trabalho.

RESUMO

O *Traveling Tournament Problem* (TTP), ou Problema de Geração de Escala de Jogos para Torneios Esportivos, é um problema de otimização que trata algumas características de torneios esportivos, tendo como objetivo a minimização das distâncias percorridas pelos times no decorrer da competição. O presente trabalho apresenta o uso de novas técnicas heurísticas híbridas para a resolução da versão espelhada do TTP, utilizando um algoritmo evolutivo, chamado *Evolutionary Clustering Search* (ECS), bem como uma adaptação deste, chamado *Clustering Search* (*CS), onde a metaheurística *Variable Neighborhood Search* (VNS), será utilizada como alternativa ao algoritmo evolutivo empregado no ECS. Apresenta-se ainda, uma modelagem inédita para o método evolutivo utilizado através de uma codificação genética compacta associada a um algoritmo de expansão de código que tem por objetivo decodificar cromossomos em escalas de jogos. A validação dos resultados foi realizada em instâncias existentes na literatura e em problemas reais (Campeonato Brasileiro de Futebol). Quando possível, os resultados apresentados foram comparados com os de outros métodos já utilizados na literatura.

NEW HEURISTICS FOR THE TRAVELING TOURNAMENT PROBLEM

ABSTRACT

The *Traveling Tournament Problem* (TTP) is an optimization problem that represents some types of sports timetabling, where the objective is to minimize the total distance traveled by the teams. This work presents the use of hybrid heuristics to solve the mirrored TTP, using an evolutionary algorithm, called *Evolutionary Clustering Search* (ECS) and an adaptation of this, called *Clustering Search* (*CS), where the metaheuristic *Variable Neighborhood Search* (VNS) was used instead of the evolutionary algorithm of the ECS. It presents the use of Genetic Algorithm with a compact genetic codification in conjunction with an algorithm to expand the code. The validation of the results were done in benchmark problems available in literature and real benchmark problems, e.g. Brazilian Soccer Championship.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

*CS	–	*Clustering Search*
2-Opt	–	*2-Optmal*
AA	–	Analisador de agrupamentos
ACC	–	*Atlantic Coast Conference*
AE	–	Algoritmo evolutivo
AG's	–	Algoritmos Genéticos
AI	–	Agrupador iterativo
AO	–	Algoritmo de otimização local
BOX	–	*Block Order Crossover*
CBF	–	Confederação Brasileira de Futebol
CIRCn	–	Instâncias circulares
CONn	–	Instâncias constantes
DDR	–	*Double Round Robin*
ECS	–	*Evolutionary Clustering Search*
GA-SA	–	Algoritmo híbrido: Algoritmos Genéticos e *Simulated Annealing*
GRASP	–	*Greedy Randomized Adaptive Search Procedure*
ILS	–	*Iterated Local Search*
ILS-AO	–	Utilização da técnica ILS como componente AO do ECS e *CS
MDRR	–	*Mirrored Double Round Robin*
ME	–	Metaheurística
MLB	–	*Major League Baseball*
mTTP	–	*Mirrored Traveling Tournament Problem*
NLn	–	Instâncias da *National League*
PE	–	Probabilidade de Eliminação
PJCB	–	Programação de Jogos do Campeonato Brasileiro de Futebol
RNA	–	Método Randômico Não Ascendente
SA	–	*Simulated Annealing*
SLSP	–	*Sports League Scheduling Problem*
SRR	–	*Simple Round Robin*
TJ	–	Troca Jogos
TMC	–	Troca Mando de Campo
TPR	–	Troca Parcial de Rodadas
TR	–	Troca Rodadas
TT	–	Troca Times
TTP	–	*Traveling Tournament Problem*
VND	–	*Variable Neighborhood Descent*
VNS	–	*Variable Neighborhood Search*
VNS-CS	–	*Clustering Search* implementando o VNS como componente ME

LISTA DE SÍMBOLOS

n	–	Número de times participantes do torneio
D	–	Matriz simétrica de distâncias
T_i	–	Time de índice i
r_k	–	Rodada de índice k
(i, k)	–	Oponente i de T_i na rodada k
$\{T_i, T_j\}$	–	Jogo entre os times T_i e T_j
Δ	–	Variação no valor da função de avaliação usada no método SA
T	–	Temperatura corrente do SA
T_0	–	Temperatura inicial do SA
$SAmax$	–	Número máximo de iterações por valor de temperatura
C	–	Conjunto de restrições associadas ao TTP
$N^{(k)}$	–	Representação da k-ésima vizinhança do VNS
f	–	Função que avalia a qualidade de um indivíduo
$\|P\|$	–	Tamanho da população corrente
$\|P_0\|$	–	Tamanho da população inicial
s	–	Indivíduo
c	–	Centro de uma região, possivelmente, promissora
r	–	Raio de uma região, possivelmente, promissora
b	–	Estratégia de busca associada aos *clusters*
s'	–	Solução vizinha da solução corrente s
s''	–	Ótimo local obtido a partir do visinho s'
w_i	–	Peso associado à restrição de índice j
$f_j(s)$	–	Função de avaliação
D_i	–	Deslocameto total do time de índice i
$iter$	–	Contador de iterações
$IterMax$	–	Número máximo de iterações
k	–	K-ésima estrutura de vizinhança dos métodos VND e VNS
nv	–	Número de estruturas de vizinhança diferentes
s_i	–	Solução inicial do método *Path-Relinking*
s_g	–	Solução guia do método *Path-Relinking*
V	–	Vetor de times
N_i	–	Número de jogos consecutivos dentro ou fora de casa
PE_i	–	Probabilidade de eliminação de um indivíduo de índice i
F_i	–	Fatia proporcional ao PE_i de um indivíduo de índice i na roleta
NC_{max}	–	Número máximo de *clusters* do ECS e *CS
λ_t	–	Densidade que indica se um *clusters* pode ser considerado promissor
PD	–	Sensibilidade relacionada a densidade dos *clusters*
NS	–	Número de indivíduos que são selecionados a cada geração
NT_c	–	Número total de *clusters*
α	–	Parâmetro utilizado para relaxar o critério de aceitação do ILS
C_{ativo}	–	Marca se um *cluster* está ativo
CP	–	Marca um *cluster* promissor

1 INTRODUÇÃO

O escalonamento de jogos esportivos, ou programação de jogos, vem se tornando nos últimos anos uma das classes mais importantes de problemas combinatórios. Para as aplicações de Pesquisa Operacional o gerenciamento deste tipo de atividade esportiva é uma área bastante promissora e ainda pouco explorada. A quantidade de problemas potenciais gerados pela organização das competições esportivas é muito grande, pois estas envolvem vários aspectos logísticos e econômicos. O tratamento destes aspectos de forma conjunta é uma tarefa de grande importância, pois as competições esportivas representam uma das maiores atividades econômicas ao redor do mundo. Para a maioria das competições (tais como futebol, futsal, voleibol, basquetebol, hóquei), onde os jogos são disputados entre dois times e realizados em vários locais ao longo de um determinado período de tempo, há a necessidade de se fazer um bom escalonamento dos jogos, ou seja, gerar escalas que atendam às restrições definidas pela entidade organizadora.

Além da necessidade de se escalonar jogos, alguns fatores fortalecem a aplicação de técnicas de otimização em problemas desta natureza, tais como: os times e seus patrocinadores não querem perder seus investimentos em jogadores e infra-estrutura em conseqüência de escalonamentos mal realizados ou mal organizados; as competições esportivas representam significantes fontes de renda das redes de rádio e televisão mundial, sendo estas as principais patrocinadoras das competições; as escalas de jogos interferem diretamente no desempenho dos times no decorrer da competição; entre inúmeros outros fatores.

Olhando o problema pelo lado dos estudiosos da Pesquisa Operacional, o principal atrativo é que as competições esportivas geram problemas de otimização extremamente desafiadores, além de alcançar grande difusão nos meios de comunicação.

1.1 Introdução ao Problema

O problema de geração de escalas de jogos para competições esportivas é apresentado na literatura como *Traveling Tournament Problem*, da classe de problemas conhecida como *Sports Timetabling*.

Basicamente, a tarefa de gerar uma escala de jogos consiste em fazer com que todo time participante da competição confronte no mínimo uma vez todos os outros (condição que varia entre as competições) e que todos os times joguem em todas as rodadas com oponentes diferentes (seja em sua cidade sede ou fora dela). Esta tarefa representa apenas uma parte do planejamento de uma competição esportiva, já que a escala determinada para os jogos não define as regras nem o formato da competição.

Problemas combinatórios dessa natureza contêm em geral muitas restrições conflitantes que devem ser satisfeitas, e diferentes objetivos a cumprir, como por exemplo, a minimização dos deslocamentos dos times durante o campeonato, realização de apenas uma partida por time e por dia de jogo, realização de determinados jogos em estádios e em datas pré-estabelecidas, número mínimo de partidas consecutivas realizadas na cidade sede do time e fora dela, entre outros.

A dificuldade dos problemas de programação de jogos cresce quando as cidades sedes do times que participam das competições estão distribuídas, geograficamente, em grandes regiões. No caso do Campeonato Brasileiro de Futebol (organizado pela Confederação Brasileira de Futebol - CBF), por exemplo, participam equipes da região sul à região norte do Brasil. Uma simples viagem entre Porto Alegre e Belém demora quase um dia inteiro, uma vez que não existem vôos diretos entre essas cidades. Portanto, minimizar a distância viajada pelas equipes torna-se um objetivo muito importante a ser considerado, para diminuir os gastos com transporte e permitir um tempo maior de descanso aos jogadores.

A geração de escalonamentos satisfatórios, respeitando essas condições e objetivos, é um problema de difícil solução. A dificuldade de solução desse problema é atribuída, principalmente, ao grande número de possibilidades a serem analisadas. Em outras palavras, a geração de escalas de jogos apresenta uma explosão combinatória de soluções candidatas, fazendo com que o processo de busca exaustiva pela solução ótima represente um procedimento computacional intratável.

De acordo com Concílio e Zuben (2002), para uma competição envolvendo n times confrontando-se entre si em turnos completos (todos os times jogando em todas as rodadas, portanto assume-se que n é par), o número de combinações possíveis é dado pela seguinte fórmula:

$$(n-1)!(n-3)!(n-5)!...(n-(n-1))! \times 2^{(n-1)\times\frac{n}{2}} \tag{1.1}$$

Exemplificando a magnitude do espaço de soluções, para uma competição com 20 times participantes há $2,9062 \times 10^{130}$ combinações possíveis.

Trata-se, portanto, de um problema de grande importância e interesse prático, uma vez que uma boa escala afeta não apenas os resultados dos jogos, mas também todo rendimento financeiro da competição. Conforme descrito anteriormente, conseguir um resultado satisfatório é uma tarefa muito difícil, mesmo quando realizada por um especialista ou

pela utilização de ferramentas convencionais.

1.2 Técnicas de Resolução

Estudos vêm sendo realizados por diversos autores (Easton *et al.* (2001), Benoist *et al.* (2001), Zhang (2002), Miyashiro *et al.* (2002), Biajoli *et al.* (2003b), Anagnostopoulos *et al.* (2003), Ribeiro e Urrutia (2004), Henz *et al.* (2004), entre outros) para tentar resolver esses problemas, apresentando uma variedade de abordagens: programação inteira, programação linear, programação por restrições, heurísticas híbridas, etc. Porém, por ser pertencente a classe de problemas NP-difíceis, este problema é normalmente abordado por técnicas heurísticas de solução, pois o tratamento do mesmo através de modelos exatos está limitado a problemas de pequenas dimensões.

Entende-se como sendo heurísticas as técnicas que exploram o espaço de busca a procura de boas soluções, próximas ou não da solução ótima e com custo computacional razoável. A grande desvantagem das heurísticas convencionais é que, ao encontrarem um ótimo local, terminam sua execução e aceitam este resultado como solução para o problema. Pode-se citar os seguintes exemplos de métodos heurísticos convencionais: Método de Descida, Método Randômico de Descida e Método Randômico Não Ascendente - RNA.

A necessidade de escapar de ótimos locais, e assim explorar um número maior de soluções, deu origem a uma classe de técnicas mais robustas, chamadas metaheurísticas. As metaheurísticas foram criadas, principalmente, para solucionar problemas de otimização combinatória considerados NP-difíceis, ou seja, de difícil resolução exata, exceto para problemas de pequenas dimensões. São procedimentos destinados a encontrar soluções de boa qualidade, eventualmente a solução ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico (RIBEIRO, 1996). As metaheurísticas utilizam buscas locais para obtenção das soluções, e contrariamente às heurísticas convencionais podem escapar de ótimos locais. Dentre as metaheurísticas existentes atualmente pode-se citar como sendo as mais relevantes: *Simulated Annealing* (KIRKPATRICK *et al.*, 1983; DOWSLAND, 1993), Busca Tabu (GLOVER, 1989; GLOVER, 1990; GLOVER; LAGUNA, 1993), Algoritmos Genéticos (GOLDBERG, 1989), *Greedy Randomized Adaptive Search Procedure* - GRASP - (FEO; RESENDE, 1995), Colônia de Formigas (DORIGO *et al.*, 1996) e *Variable Neighborhood Search* - VNS (MLADENOVIC; HANSEN, 1997).

1.3 Objetivos do Trabalho

O presente trabalho pretende contribuir com o desenvolvimento do método *Evolutionary Clustering Search* (ECS), proposto por Oliveira (2004), utilizando-o na resolução do

Traveling Tournament Problem, em sua versão espelhada. O ECS combina algoritmos evolutivos com técnicas de agrupamentos para descobrir áreas de busca mais promissoras. A estratégia de busca fica mais agressiva quando tais áreas são encontradas, aplicando nestas uma busca local. Espera-se com isso uma convergência mais rápida por parte do algoritmo, acarretando em possível diminuição do esforço computacional, uma vez que a busca local é realizada de forma mais racional. Será verificado ainda o comportamento do método *CS, que é uma adaptação do ECS, onde a metaheurística *Variable Neighborhood Search* (VNS) será utilizada como alternativa ao algoritmo evolutivo do ECS.

Define-se ainda uma modelagem para a abordagem evolutiva que trata de forma inédita o TTP ao propor uma codificação genética compacta associada a um algoritmo de expansão de código, que tem por objetivo decodificar cromossomos em escalas de jogos.

A validação dos métodos será realizada através de instâncias existentes na literatura, definidas por Easton *et al.* (2001), e adaptadas para a versão espelhada por Ribeiro e Urrutia (2004). Serão utilizadas também algumas instâncias de problemas reais (Campeonato Brasileiro de Futebol). Quando possível, os resultados apresentados serão comparados com os de outros métodos já utilizados.

1.4 Organização da Dissertação

Esta dissertação está organizada em 7 capítulos, sendo esta introdução o primeiro.

No Capítulo 2, apresenta-se a descrição completa do *Traveling Tournament Problem* (EASTON *et al.*, 2001) e suas variantes: a versão espelhada do TTP, ou *Mirrored Traveling Tournament Problem - mTTP* (RIBEIRO; URRUTIA, 2004) e o Problema de Programação de Jogos do Campeonato Brasileiro de Futebol (BIAJOLI *et al.*, 2003b).

No Capítulo 3 é realizada uma revisão bibliográfica sobre algumas das várias técnicas que vêm sendo utilizadas para resolução do problema de programação de jogos de torneios esportivos. É apresentada a forma como diversos autores tratam o problema.

No Capítulo 4, são apresentadas as técnicas de resolução que foram utilizadas neste trabalho.

No Capítulo 5, é definida formalmente a metodologia adotada, partindo-se da modelagem e representação do problema até a forma como as técnicas foram empregadas.

No Capítulo 6, apresentam-se os experimentos computacionais realizados, bem como os resultados obtidos.

E por último, no Capítulo 7, apresentam-se as conclusões e considerações finais sobre o trabalho desenvolvido.

2 DESCRIÇÃO DO PROBLEMA

Neste capítulo é descrito o *Traveling Tournament Problem* (TTP) e suas variantes, a saber: *Mirrored Traveling Tournament Problem* (mTTP) e o Problema de Programação de Jogos do Campeonato Brasileiro de Futebol (PJCB). Serão apresentadas as descrições formais e restrições associadas a cada uma dessas classes.

O presente trabalho tem seu foco no problema espelhado, ou seja, o *Mirrored Traveling Tournament Problem*, por dois principais motivos. Em primeiro lugar pelo fato de existirem instâncias públicas que vêm sendo tratadas por diversos autores, permitindo assim analisar a eficiência do método empregado. Em segundo lugar, e não menos importante, o fato deste apresentar algumas características similares a torneios comuns na América Latina.

2.1 *Traveling Tournament Problem*

O *Traveling Tournament Problem* (TTP) é um problema de escalonamento de competições esportivas com fortes componentes de otimização, uma vez que apresenta inúmeras restrições que possuem uma dependência muito grande entre si. Ele foi proposto inicialmente por Easton *et al.* (2001), sendo uma abstração do problema real da MLB - *Major League Baseball* - dos Estados Unidos.

"Dados n times (n par), uma competição *double round robin* (DRR) é um conjunto de jogos onde todo time joga com os outros exatamente duas vezes, sendo uma vez em sua cidade sede e uma fora dela. Um jogo é especificado e ordenado em pares de oponentes. Exatamente $2(n-1)$ rodadas são requeridas para jogar um *double round robin*. As distâncias entre as cidades sedes dos times são dadas por uma matriz D, n x n. Cada time inicia o torneio em sua cidade sede e começa a viajar para cumprir seus jogos nas sedes escolhidas. Cada time retorna (se necessário) para sua cidade sede no fim de um ciclo de jogos", (adaptado de Easton *et al.* (2001)).

A restrições associadas ao TTP são:

1) " *Double round robin*"(cada time joga duas vezes com cada um dos outros, por exemplo, A x B e B x A): n times necessitam $2(n-1)$ rodadas;

2) Cada time não pode participar de mais de três partidas consecutivas dentro ou fora de suas cidades sedes;

3) Jogos entre os mesmos times não podem ser consecutivos (A x B seguido imediatamente por B x A);

4) Deve-se minimizar o somatório das distâncias de viagem dos times participantes (todos os times iniciam em suas cidades sedes e retornam a ela no fim do torneio);

5) As instâncias NLx pegam as x primeiras cidades na matriz de distâncias (NLx representam as instâncias contidas em NL para as x primeiras cidades, sendo $x = 4, 6, 8, 10, 12, 14$ e 16 - instâncias disponíveis em TOURN *website*).

Em relação às restrições citadas acima, alguns aspectos devem ser ressaltados. Primeiramente, a distância entre a cidade sede de um time A para a cidade sede de um time B é a mesma distância da cidade sede de B para a de A, ou seja, a distância entre os times é dada numa matriz D simétrica n x n, onde n é o número de times. Deve-se assumir que os times devem estar em suas cidades sedes no início o no fim do torneio. Finalmente, a distância de viagem dos times é o somatório das distâncias entre as sedes dos times alocados aos jogos de cada rodada.

2.2 *Mirrored Traveling Tournament Problem*

O *Mirrored Traveling Tournament Problem* (mTTP) é uma variante do problema original, ao qual foi adicionado uma restrição. Ele foi proposto por Ribeiro e Urrutia (2004) e apresenta, em alguns aspectos, similaridades com torneios de futebol muito populares na América Latina (como por exemplo o Campeonato Brasileiro de Futebol). Dentre estas similaridades pode-se citar como sendo a principal, a existência do conceito de turno e returno. Abaixo segue a descrição formal do mTTP.

Em torneios simples, *simple round robin* (SRR), disputados por n times (n par), cada time joga com cada um dos outros exatamente uma vez em $n-1$ rodadas. Já em torneios do tipo *double round robin* (DRR) cada time joga com cada um dos outros duas vezes, sendo um jogo em casa e outro fora de casa, sem distinção de turnos. Um *Mirrored Double Round Robin* (MDRR) é uma das variações do TTP, definida recentemente como *Mirrored Traveling Tournament Problem* (mTTP). O mTTP pode ser considerado um SRR 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). Isto quer dizer que, se um time A jogou em casa contra um time B na primeira rodada do turno, ele deverá jogar com o mesmo time B na primeira rodada do returno, porém fora de sua cidade sede (Turno: A x B; Returno: B x A).

Para este tipo de torneio, devemos atender às mesmas restrições do TTP, com uma adicional: cada jogo deve ser realizado apenas uma vez em cada turno. Dessa forma, a restrição que trata a não repetição de um jogo dentro do mesmo turno sempre será atendida. Deve-se cumprir o mesmo objetivo de minimização das distâncias percorridas,

levando em consideração as particularidades do problema.

2.3 Programação de Jogos do Campeonato Brasileiro de Futebol

O Problema de Programação de Jogos do Campeonato Brasileiro de Futebol (PJCB) pode ser considerado uma versão específica do mTTP, pois apresenta algumas restrições adicionais relativas à realidade nacional e também às especificações da entidade organizadora, no caso a CBF - Confederação Brasileira de Futebol. O primeiro trabalho que trata o Campeonato Brasileiro de Futebol com todas as suas restrições foi desenvolvido por Biajoli *et al.* (2003b), obtendo melhoras significativas em relação às escalas oficiais praticadas nos anos de 2003 e 2004.

Desde 2003 o Campeonato Brasileiro de Futebol está sendo organizado como um torneio de dois turnos, onde os times participantes se enfrentam duas vezes, sendo um jogo em casa e outro fora de casa, realizados em turnos diferentes. Os seguintes requisitos definidos pela CBF para o PJCB devem ser satisfeitos:

1) Um time não pode jogar mais de uma vez na mesma rodada;

2) O campeonato deve ser realizado em turno e returno, isto é, cada time deve jogar duas vezes com cada um dos outros times durante o campeonato, sendo os jogos realizados em sedes e turnos diferentes;

3) Nas duas primeiras rodadas de cada turno os jogos devem ser realizados de forma alternada, ou seja, um jogo dentro de casa e outro fora de casa;

4) As duas últimas rodadas de cada turno devem ter a configuração inversa das duas primeiras em relação ao mando de campo. Por exemplo, se os dois primeiros confrontos de um time foram jogar fora de casa e depois em casa, respectivamente, então os dois últimos confrontos desse time devem ser, respectivamente, jogar em casa e depois fora de casa;

5) Não pode haver jogos entre clubes do mesmo estado na última rodada (clássicos estaduais);

6) A diferença entre o número de jogos realizados fora de casa e dentro de casa deve ser no máximo um dentro de cada turno;

7) Nenhum time deve jogar mais de duas vezes consecutivas dentro de casa ou fora de casa;

8) As rodadas do returno devem ter a mesma configuração das rodadas do turno, porém com os mandos de campo invertidos;

Além do atendimento destas restrições, procura-se atender dois objetivos. O primeiro e principal é a minimização da distância total de deslocamento dos times no decorrer do campeonato. O segundo consiste em minimizar a diferença entre o time que mais se deslocou para o time que menos se deslocou no campeonato, de forma a manter o equilíbrio entre os deslocamentos dos times.

Alguns trabalhos vêm sendo desenvolvidos na tentativa de divulgar o uso de técnicas de otimização na resolução de problemas reais como o do Campeonato Brasileiro de Futebol, para que os organizadores deste evento e de outros eventos esportivos reconheçam sua importância e possam aplicá-los em suas competições (BIAJOLI *et al.*, 2003b; BIAJOLI *et al.*, 2003a; BIAJOLI *et al.*, 2004; MINE *et al.*, 2005; RIBEIRO; URRUTIA, 2006b).

3 REVISÃO BIBLIOGRÁFICA

O *Traveling Tournament Problem* (TTP) foi proposto inicialmente por Easton *et al.* (2001) e inserido na literatura como sendo um problema da classe *sports timetabling*, que abstrai certas características dos problemas de escalonamento de torneios reais. A idéia geral proposta pelos autores pode ser resumida da seguinte forma: a partir de um número n de times e das distâncias entre as suas cidades sedes o TTP consiste em gerar um escalonamento tal que cada time jogue exatamente duas vezes com cada um dos outros não havendo repetição de jogos, nenhum time jogue mais do que três vezes consecutivas dentro ou fora de sua sede e minimizando as distâncias de viagem total dos times no decorrer da competição (adaptado de Easton *et al.* (2001)).

A necessidade de automatização da tarefa de gerar escalas de jogos para competições esportivas vem recebendo atenção considerável nos últimos anos, visto que estas aplicações envolvem rendas significantes e geram problemas desafiadores de otimização combinatória (ANAGNOSTOPOULOS *et al.*, 2003).

Segundo Miyashiro *et al.* (2002), construir uma tabela para uma competição esportiva é uma importante tarefa para os organizadores, pois a tabela afeta não apenas os resultados dos jogos, mas também o rendimento da competição. Visto que a construção manual de uma tabela que atenda a todos os requisitos da competição e que possa ser implantada é muito difícil, a demanda por meios de escalonamentos automatizados vem crescendo, gerando cada vez mais interesse por parte da comunidade científica.

Em geral, problemas de escalonamento contêm muitas restrições conflitantes entre si para satisfazer e diferentes objetivos para se otimizar, como a minimização da distância total percorrida pelos times (BEAN; BIRGE, 1980), a realização de apenas um jogo por dia, estádios indisponíveis em datas particulares, número mínimo de dias entre jogos em casa e seus correspondentes fora de casa (HAMIEZ; HAO, 2001).

O tratamento de problemas de otimização com este grau de complexidade é conseguido caso se abra mão da solução ótima, em prol da obtenção de uma solução de boa qualidade. Como muitas vezes não se tem condição de medir o quão distante da solução ótima se encontra a solução obtida, a qualidade desta é medida em relação às soluções candidatas anteriormente produzidas pelo processo iterativo, a partir de uma condição inicial (CONCíLIO, 2000).

Esta dificuldade de se chegar à solução ótima faz com que problemas deste tipo sejam normalmente abordados por técnicas heurísticas de solução. Vários autores, em diferentes contextos, estão trabalhando na resolução de problemas de escalonamento em diversos

tipos de esportes, tais como futebol (Concílio e Zuben (2002), Biajoli *et al.* (2003b), Biajoli *et al.* (2003a), Biajoli *et al.* (2004)), basquete (Easton *et al.* (2001)), baseball (Yang *et al.* (2002), Henz (2001)), cricket (Armstrong e Willis (1993), Willis e Terril (1994)), entre outros. Em Easton *et al.* (2004) apresenta-se um *survey* sobre os trabalhos que vêm sendo desenvolvidos. Atualmente, este é o *survey* mais completo dos trabalhos produzidos na área nos últimos 35 anos, listando 45 publicações com diversos problemas esportivos como referência.

Um dos primeiros trabalhos desenvolvidos utilizando conceitos de computação evolutiva aplicados a problemas de escalonamento em esportes foi o de Costa (1995), onde um método híbrido combinando as técnicas Algoritmos Genéticos e Busca Tabu foi proposto para resolver um problema de escalonamento dos jogos da Liga Americana de Hóquei. Idéias de Inteligência Artificial foram introduzidas para guiar o processo de evolução natural, como por exemplo, a fase de mutação do Algoritmo Genético foi substituída por uma busca local guiada pelo método Busca Tabu.

Nemhauser e Trick (1998) tratam o problema de escalonamento de jogos de um torneio de basquete (ACC - *Atlantic Coast Conference*) envolvendo nove times de diferentes localidades dos Estados Unidos utilizando uma combinação de uma técnica de programação inteira e uma técnica de enumeração de soluções. O procedimento proposto contém três fases, sendo que a primeira fase consiste em gerar um conjunto de padrões de cardinalidade igual ao número n de times. Cada padrão consiste numa seqüência de $(n-1)$ letras, que representam o mando de campo ("C" para casa ou "F" fora). Na segunda fase os jogos são atribuídos aos padrões, sem o estabelecimento de quais times disputam cada jogo. Na terceira e última fase, os times são então atribuídos aos padrões, gerando uma escala de jogos. As duas primeiras fases são resolvidas por técnicas de programação inteira, enquanto a última é resolvida por uma técnica de enumeração implícita de soluções.

Trick (2001) apresenta um método, também de programação inteira, porém com duas fases para resolver este mesmo problema. Na primeira fase, os times são alocados ignorando-se os requisitos de mando de campo. Somente na segunda fase esses requisitos são observados, sendo respeitado o limite estabelecido de jogos consecutivos dentro e fora das cidades sedes dos times.

Concílio (2000) aborda o problema da montagem da tabela de jogos do Campeonato Paulista de Futebol de 1997 utilizando programação genética. Foi proposta uma aplicação conjunta de computação evolutiva através de uma representação compacta dos indivíduos, busca local utilizada na recuperação de viabilidade e técnicas de otimização baseadas em restrições.

Hamiez e Hao (2001) apresentam um método baseado na metaheurística Busca Tabu para resolver o SLSP - *Sports League Scheduling Problem* - descrito por McAloon *et al.* (1997), e que é uma das versões existentes de problemas de escalonamento em esportes. O algoritmo desenvolvido (TS-SLSP) mostrou ser bastante robusto ao obter resultados iguais e em alguns casos superiores em tempo de processamento menores que outras metodologias empregadas até aquele momento, tais como programação por restrições, programação linear e buscas locais simples.

Yang *et al.* (2002) tratam o problema da MLB - *Major League Baseball* - dos Estados Unidos usando uma estratégia evolutiva associada a dois métodos de busca local, *Simulated Annealing* e 2-Opt respectivamente. A idéia consiste em utilizar o Algoritmo Genético aplicando vários operadores de mutação nos indivíduos e, os métodos de busca local citados acima para melhorias individuais a cada geração.

Biajoli *et al.* (2003b) aplica, ao problema de programação de jogos do Campeonato Brasileiro de Futebol, a técnica *Simulated Annealing* (SA) em duas versões diferentes da competição. Na primeira versão, disputada até o ano de 2002, o campeonato foi disputado em turno único. Nesta foi aplicada à técnica SA pura (BIAJOLI *et al.*, 2003a). Na segunda versão o campeonato foi disputado em turno e returno (BIAJOLI *et al.*, 2004). Para resolução de tal problema, usou-se uma metodologia híbrida, onde três técnicas foram aplicadas. Primeiramente o SA foi aplicado em duas etapas. Na primeira etapa o objetivo era obter uma escala viável, atendendo todas as restrições essenciais do problema. Na segunda etapa o algoritmo era executado com o objetivo de obter uma boa atribuição dos mandos de campos aos jogos, obedecendo às restrições não essenciais, quando possível, e minimizando a distância total de viagem dos clubes, bem como a diferença entre a distância percorrida pelo time que mais se deslocou para o que menos se deslocou. Após a execução do SA foram aplicadas mais duas técnicas com o objetivo de refinar a melhor solução encontrada até então. Tal refinamento foi efetuado pelo algoritmo Busca Tabu, através da troca de mandos de jogos escolhidos aleatoriamente, seguido por uma heurística de refinamento da solução por enumeração restrita de tabelas simétricas (BIAJOLI *et al.*, 2003a; BIAJOLI *et al.*, 2004). Nesta fase era realizada a troca de dois times escolhidos aleatoriamente, gerando uma nova tabela simétrica a anterior, mas com rotas de viagem diferentes. A enumeração era realizada de forma restrita pelo fato de haver um número muito elevado de soluções. Por exemplo, o número de soluções diferentes que podem ser obtidas a partir de um conjunto de 24 clubes, onde 6 são de São Paulo, 3 do Rio de Janeiro, 3 do Rio Grande do Sul, 3 do Paraná, 2 de Minas Gerais, 2 da Bahia , 2 de Santa Catarina, 1 de Belém, 1 do Ceará e 1 de Goiás (Clubes participantes da edição do Campeonato Brasileiro de Futebol de 2003), é dado pela fórmula da Permutação com Repetição:

$$P^{24}_{6,3,3,3,2,2,2} = \frac{24!}{6!3!3!3!2!2!2!} = 498.688.594.500.096.000 \quad (3.1)$$

Os resultados obtidos por esta metodologia validaram a utilização da mesma, uma vez que as escalas geradas foram bem superiores àqueles praticadas nas versões oficiais do Campeonato Brasileiro, considerando tempo de processamento computacional na ordem de poucos minutos.

Anagnostopoulos *et al.* (2003) também propuseram uma aplicação da metaheurística *Simulated Annealing* ao TTP (TTSA), porém utilizando-a nas instâncias da MLB. Como de uso, o algoritmo inicia sua execução a partir de uma solução aleatória e realiza movimentos para avaliar a vizinhança da solução corrente. São quatro as idéias principais do TTSA: 1) As restrições foram separadas em restrições essenciais e não essenciais; 2) TTSA é baseado numa ampla vizinhança, de ordem $O(n^3)$, onde n é o número de times; 3) A função objetivo do TTSA é ajustada para balancear o tempo gasto em regiões de soluções viáveis e inviáveis; e 4) é aplicado o conceito de reaquecimento para escapar de mínimos locais em temperaturas baixas. O TTSA obteve resultados expressivos, sendo ainda um dos responsáveis por boas soluções para as instâncias da MLB. O maior problema apresentado pelo TTSA é o enorme esforço computacional exigido, visto que algumas soluções foram encontradas após dias de processamento.

Ribeiro e Urrutia (2004) formalizaram uma versão espelhada do TTP, chamada *Mirrored Traveling Tournament Problem* (mTTP), sendo esta versão similar a modelos de campeonatos bastante comuns na América Latina. O mTTP está para uma abstração do Campeonato Brasileiro de Futebol assim como o TTP está para uma abstração da *Major League Baseball* (MLB), uma vez que ambos levam em consideração apenas as restrições chaves desses modelos de competição, descartando as restrições específicas de cada problema. Além da formalização do mTTP eles propuseram uma técnica híbrida para resolução do mesmo. Esta técnica é baseada na metaheurística GRASP (*Greedy Randomized Adaptative Search Procedure*) associada a ILS (*Iterated Local Search*). A técnica, entitulada GRILS-mTTP, obteve bons resultados tanto para a versão espelhada como para a versão não espelhada do TTP, exigindo pouco esforço computacional.

Outra utilização bem sucedida da metaheurística *Simulated Annealing* foi a realizada por Zhang *et al.* (2004). Neste trabalho os autores decidiram dividir o espaço de busca entre o SA, para realizar busca por escalas, e o algoritmo *Hill-Climbing*, para explorar a atribuição dos times. Entre as duas técnicas um controlador fazia o gerenciamento da execução, fixando os times e fazendo chamadas ao SA, para que o mesmo gerasse escalas melhores. A melhor escala encontrada é então passada para o *Hill-Climbing* para uma

busca por melhores atribuições de times. As atribuições de times que geraram a melhor escala são então retornadas ao SA. Este ciclo se repete até que algum critério de parada seja obedecido.

Quando os jogos de um torneio são sempre disputados nas cidades sedes dos times que se enfrentam é necessário definir-se qual time jogará em sua cidade sede e qual viajará até esta localidade. Aos times que jogam em casa, portanto, possuem o mando de campo, dá-se o nome de *mandante*, enquanto os outros times são os *visitantes*. Quando um time participa de duas partidas consecutivas como mandante ou visitante, dizemos que este time tem uma *parada*, ou *quebra*. Neste sentido, Werra (1980) mostrou como gerar escalas com número mínimo de paradas, além de provar outras propriedades dos problemas de geração de escalas de torneios esportivos utilizando teoria de grafos. Elf *et al.* (2003) tratam o problema de minimização das quebras quando a seqüência dos jogos já está definida, faltando apenas a determinação do mandos de campos. Miyashiro *et al.* (2002) caracterizam completamente as escalas com número mínimo de paradas para torneios SSR com até 26 times participantes.

Outra contribuição realizada por Ribeiro e Urrutia (2006a), investiga a relação entre a distância percorrida pelos times e as paradas (uma parada equivale a dois jogos consecutivos dentro ou fora da sede do time). Neste trabalho eles provam que o problema de maximização das paradas e o problema de minimização da distância percorrida pelos times são equivalentes, dentro de uma nova classe de instâncias constantes, também definidas por eles. Uma instância constante é aquela onde a distância entre as cidades sedes do times é sempre igual a 1. Esta conexão entre maximização das paradas e minimização das distâncias é utilizada para derivar *lower bounds* para o mTTP e com isso provar a otimalidade das soluções encontradas por heurísticas.

Hentenryck e Vergados (2006) apresentam uma adaptação da técnica TTSA de Anagnostopoulos *et al.* (2003) para que a mesma contemple também as instâncias espelhadas do TTP. Tal adaptação consistiu, basicamente, na inclusão das restrições referentes a manutenção do turno espelhado. Além dessa adaptação, este trabalho apresenta dois novos movimentos, chamados de *ReverseAwayRun* e *ReverseAwayRunMirrored*. A idéia principal destes movimentos é trocar o mando de campo de um subconjunto de jogos realizados fora de casa, na tentativa de minimizar as ocorrências dos pares de jogos {fora, fora}, {casa, fora} e {fora, casa}, pois estes pares geram viagens. Com isso, é possível preservar boa parte da seqüência de jogos dos times minimizando as viagens dos mesmos. Esse trabalho, apresentou novos e melhores resultados para 12 das 21 instâncias testadas. Assim como em sua primeira versão, o maior problema do TTSA continuou sendo o enorme esforço computacional exigido, levando dias

de processamento para encontrar algumas soluções.

4 MÉTODOS DE RESOLUÇÃO

Diversos problemas comuns no nosso dia a dia podem ser modelados e resolvidos como problemas de otimização combinatória. Problemas de geração de rotas, ou roteamento, escalonamento de jogos, programação de quadro de horários, localização de facilidades, atribuição, empacotamento, entre outros, fazem parte dessa gama enorme de problemas que são resolvidos por técnicas de otimização.

Uma forma simples de se resolver tais problemas seria enumerar todas as soluções possíveis e no final escolher a melhor. Porém, em sua grande maioria, a resolução nem sempre é realizada de forma tão fácil. Isso porque muitos desses problemas se encontram na classe de problemas NP-difíceis, isto é, não se conhece nenhum algoritmo que, em tempo polinomial, consiga resolvê-los de forma exata, exceto para problemas de pequenas dimensões. Em problemas reais o que geralmente acontece é uma explosão combinatória muito grande, tornando-se impraticável a enumeração de todas as soluções. Portanto, o uso de métodos exatos e determinísticos em problemas dessa natureza se torna bastante restrito. Em contrapartida, o que geralmente acontece na prática é que soluções de boa qualidade, ao invés de ótimas, são suficientes para muitos problemas.

Este motivo tem levado os pesquisadores a concentrar esforços em métodos de busca local na resolução de problemas com este grau de complexidade, uma vez que, tais métodos, realizam uma exploração mais inteligente no espaço de busca, exigindo pouco esforço computacional.

Neste trabalho alguns desses métodos serão explorados com o objetivo de se obter, em tempo reduzido, soluções tão próximas quanto possíveis da solução ótima. Os métodos que serão abordados já foram aplicados de forma eficaz na resolução de diversos outros problemas de otimização. A seguir serão apresentadas breves descrições de tais métodos.

4.1 Métodos de Busca Local

Métodos de busca local são técnicas baseadas na noção de vizinhança e aplicadas em problemas de otimização. De forma mais objetiva, seja S o espaço de pesquisa de um dado problema de otimização e f a função objetivo a minimizar. Uma função N, dependente da estrutura do problema tratado, associa a cada solução viável $s \in S$, sua vizinhança $N(S) \subseteq S$. Cada solução s' de $N(S)$ é chamada de vizinho de s. Denomina-se movimento a modificação m que transforma uma solução s em uma nova solução, s', que esteja em sua vizinhança. Tal transformação é representada por $s' \leftarrow s \oplus m$.

Podemos dividir as técnicas de busca local em dois tipos diferentes: heurísticas

(convencionais ou construtivas) e metaheurísticas. As heurísticas convencionais são métodos de busca local que, ao encontrarem um ótimo local, terminam sua execução e aceitam este resultado como solução para o problema. As heurísticas construtivas são normalmente desenvolvidas para resolver um problema específico.

Os Métodos Metaheurísticos são métodos de busca local que possuem certas características que os tornam mais robustos. Com isso esses métodos possuem mais chances de escapar de ótimos locais. Em níveis de hierarquia, uma metaheurística é uma heurística que faz uso de outra heurística subordinada, com certo grau de flexibilidade para possibilitar a fácil adaptação a uma grande variedade de problemas.

De forma generalizada, uma técnica de busca local, começando de uma solução inicial s_0 (a qual pode ser gerada de forma aleatória ou obtida através do emprego de alguma outra técnica), navega pelo espaço de pesquisa, passando, iterativamente, de uma solução para outra que seja sua vizinha.

4.2 Heurísticas

Os algoritmos heurísticos são desenvolvidos com o objetivo de resolver problemas de otimização com grau de complexidade elevado, onde a resolução exata se torna inviável, ou até mesmo impraticável.

Basicamente, as heurísticas são algoritmos que exploram o espaço de busca à procura de soluções de boa qualidade em tempo computacional razoável. Porém, muitas vezes, não se tem condições de saber o quão distante da solução ótima se encontra a solução obtida, fazendo com que a qualidade desta seja medida em relação às soluções candidatas anteriormente produzidas pelo processo iterativo, a partir de uma condição inicial.

As heurísticas construtivas, como o próprio nome diz, são aquelas que constroem uma solução a partir de um ponto inicial vazio, adicionando-se novos elementos (arestas, variáveis, vértices) de forma iterativa, até que seja obtida uma solução viável. As heurísticas gulosas também podem ser consideradas construtivas, pois possuem este comportamento, tentando a cada iteração maximizar a melhoria local.

Heurísticas convencionais, ou de melhoria, são técnicas que seguem regras pré-estabelecidas para modificar uma dada solução inicial, gerando uma solução vizinha iterativamente até que o critério de parada seja atingido. Em geral, o critério de parada é atingido quando uma solução gerada não pode ser mais melhorada, caracterizando um ótimo local. Um ótimo local é o melhor ponto dentro de uma localidade definida pela relação de vizinhança que é induzida por uma dada heurística de busca (OLIVEIRA, 2004).

Pode-se citar como exemplos de métodos heurísticos convencionais as seguintes técnicas: Método de Descida, Método Randômico de Descida, Método Randômico Não Ascendente (RNA) e Descida 2-Opt. Neste trabalho apenas a heurística RNA foi utilizada, sendo descrita na próxima seção.

4.2.1 Método Randômico Não Ascendente

O Método Randômico Não Ascendente (RNA) é um método alternativo aos métodos de descida convencionais. Estes últimos requerem uma exploração de toda a vizinhança de uma solução corrente. Já o Método Randômico Não Ascendente consiste em analisar um vizinho qualquer, escolhido aleatoriamente, e aceitá-lo como nova solução corrente se esta solução for igual ou melhor, caso contrário a solução corrente é mantida e outro vizinho é gerado. O procedimento é executado até que um número pré-estabelecido de iterações sem melhora no valor da solução corrente é atingido.

A Figura 4.1 apresenta o pseudocódigo do Método RNA aplicado ao refinamento de uma solução s, dado um problema de minimização de uma função *f(.)*, utilizando movimentos de uma estrutura de vizinhança *N(.)*.

```
Procedimento RNA(f(.), N(.), IterMax, s)
1   Iter ← 0; {Contador de iterações sem melhora }
2   enquanto (Iter < IterMax) faça
3       Iter ← Iter + 1;
4       Selecione aleatoriamente s' ∈ N(s);
5       se (f(s') ≤ f(s)) então
6           Iter ← 0;
7           s ← s';
8       fim-se;
9   fim-enquanto;
10  Retorne s;
fim-RNA;
```

Figura 4.1 - Algoritmo *RNA*

Por esse método, entretanto, é possível caminhar pelo espaço de pesquisa através de movimentos laterais. Assim, ele tem condições de percorrer caminhos de descida que passam por regiões planas. Ou seja, se a pesquisa chega numa dessas regiões, o método tem condições de mover-se nela chegando a outra solução, diferente daquela que a ela chegou.

O método RNA é, portanto, um procedimento que executa a pesquisa no espaço de

soluções combinando movimentos de descida com movimentos laterais.

4.2.2 Método de Descida em Vizinhança Variável

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent* - VND) foi proposto por Mladenovic e Hansen (1997). É uma técnica de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções que apresentem melhora em relação à solução corrente. O método retorna à primeira estrutura de vizinhança quando uma solução melhor é encontrada.

```
Procedimento VND(f(.), N(.), r, s)
1   Seja nv o número de estruturas diferentes de vizinhança;
2   k ← 1; Tipo de estrutura de vizinhança corrente
3   enquanto (k ≤ nv) faça
4       Encontre o melhor vizinho s' ∈ N^(k)(s);
5       se (f(s') < f(s)) então
6           s ← s';
7           k ← 1;
8       senão
9           k ← k + 1;
10      fim-se;
11  fim-enquanto;
12  Retorne s;
fim-VND;
```

Figura 4.2 - Algoritmo *VND*

Dependendo do problema abordado, a busca pelo melhor vizinho (linha 4 da Figura 4.2) pode ser cara computacionalmente. Nessa situação é comum fazer a busca pela primeira solução de melhora. Outra alternativa é considerar a exploração apenas em um certo percentual da vizinhança (SOUZA, 2005).

4.2.3 *Path-Relinking*

A técnica *Path Relinking*, ou Reconexão por Caminhos, é uma estratégia de intensificação de busca, originalmente proposta por Glover (1996b), usada para explorar trajetórias que conectam soluções elites, geradas anteriormente à aplicação da técnica ou durante sua aplicação.

O processo de busca por melhores soluções consiste em, a partir de uma ou mais soluções

elites, gerar e explorar o espaço de soluções e assim chegar a outras soluções elites. Para isso, é necessário aplicar movimentos na solução corrente que possuem características da solução elite, e assim, incorporar características de boa qualidade à solução corrente.

O procedimento começa calculando a diferença simétrica entre as duas soluções, ou seja, o conjunto de movimentos necessários para se chegar à solução guia (s_g) de uma dada solução inicial (s_i). Um caminho de soluções é gerado ligando s_i e s_g. A melhor solução no caminho (s^*) é retornada pelo procedimento. A cada passo, são examinados todos os movimentos da solução corrente que ainda não foram selecionados e escolhe-se aquele que resultar na solução de menor custo. O melhor movimento é realizado, produzindo uma nova solução inicial, e o conjunto de movimentos necessários é atualizado. O procedimento termina quando s_g é alcançado, ou seja, não existe mais diferença entre s_i e s_g. A Figura 4.3 ilustra o funcionamento do *Path-Relinking*.

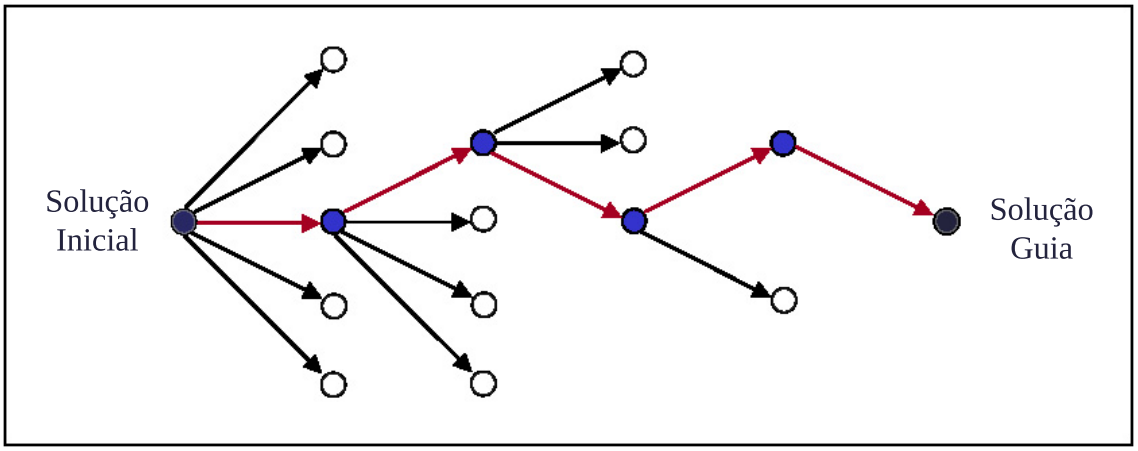

Figura 4.3 - Exemplo de movimentos do *Path-Relinking*

4.3 Metaheurísticas

As metaheurísticas foram criadas, principalmente, para solucionar problemas de otimização combinatória da classe NP-difíceis, ou seja, de difícil resolução exata, com exceção de problemas com pequenas dimensões. São procedimentos destinados a encontrar soluções de boa qualidade, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico (RIBEIRO, 1996). As metaheurísticas utilizam buscas locais para obtenção das soluções, e contrariamente às heurísticas convencionais podem escapar de ótimos locais.

As metaheurísticas, assim como as heurísticas convencionais, diferenciam entre si, basicamente, pelas seguintes características:

- Critério de escolha de uma solução inicial;
- Definição da vizinhança $N(s)$ de uma solução s;
- Critério de seleção de uma solução vizinha dentro de $N(s)$;
- Critério de parada;

Pode-se citar várias metaheurísticas existentes na literatura, entre elas, Busca Tabu, *Simulated Annealing*, *Greed Randomized Adaptative Search Procedure* (GRASP), *Variable Neighborhood Search* (VNS), Algoritmos Genéticos e outros.

4.3.1 *Simulated Annealing*

Simulated Annealing (SA) é um método de busca local que aceita movimentos de piora para escapar de ótimos locais. Ele foi proposto originalmente por Kirkpatrick *et al.* (1983), e se fundamenta em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos, operação conhecida como recozimento.

Esta técnica começa sua busca a partir de uma solução inicial qualquer. O procedimento principal consiste em um *loop* que gera aleatoriamente, em cada iteração, um único vizinho *s'* da solução corrente *s*.

A cada geração de um vizinho *s'* de *s*, é testada a variação Δ, do valor da função objetivo, isto é, $\Delta = f(s') - f(s)$. Se $\Delta < 0$, o método aceita a solução *s'*, que passa a ser a nova solução corrente *s*. Caso contrário, ou seja, $\Delta \geq 0$, a solução vizinha candidata *s'* também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, onde 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 (soluções de piora).

Inicialmente, a temperatura T assume um valor elevado T_0. Após um número fixo de iterações (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura), a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_n \leftarrow \alpha \times T_{n-1}$, sendo $0 < \alpha < 1$. Com esse procedimento, dá-se, no início uma chance maior de se escolher soluções de piora, conseqüentemente, aumenta-se à chance de se escapar de ótimos locais e, à medida que T aproxima-se de zero, o algoritmo comporta-se como o método de descida, uma vez que diminui a probabilidade de se aceitar movimentos de piora $\left(T \to 0 \Longrightarrow e^{-\Delta/T} \to 0\right)$.

O procedimento finaliza sua execução quando a temperatura chega a um valor próximo de zero e nenhuma solução que piore o valor da 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. O algoritmo básico do SA é apresentado na Figura 4.4.

Os parâmetros de controle do procedimento são a razão de resfriamento (α), o número de iterações para cada temperatura ($SAmax$) e a temperatura inicial (T_0).

```
Procedimento SA(f(.), N(.), α, SAmax, T0, s)
1   s* ← s;                  {Melhor solução obtida até então}
2   IterT ← 0;               {Número de iterações na temperatura T}
3   T ← T0;                  {Temperatura corrente}
4   enquanto (T > 0) faça
5       enquanto (IterT < SAmax) faça
6           IterT ← IterT + 1;
7           Gere um vizinho qualquer s' ∈ N(s);
8           Δ ← f(s') − f(s);
9           se (Δ < 0) então
10              s ← s'
11              se (f(s') < f(s*)) então
12                  s* ← s';
13          senão
14              Tome x ∈ [0, 1];
15              se (x < e^(−Δ/T)) então
16                  s ← s';
17          fim-se;
18      fim-enquanto;
19      T ← α × T;
20      IterT ← 0;
21  fim-enquanto;
22  s ← s*;
23  Retorne s;
fim-SA;
```

Figura 4.4 - Algoritmo *Simulated Annealing*

Fonte: Souza (2005)

4.3.2 Algoritmos Genéticos

Propostos por Holland (1975), em meados dos anos 70, os Algoritmos Genéticos (AG's) são algoritmos de otimização, baseados nos mecanismos de seleção natural e da genética, isto é, se fundamentam em uma analogia com processos naturais de evolução, nos quais, dada uma população, os indivíduos com características genéticas melhores têm maiores

chances de sobrevivência e de produzirem filhos cada vez mais aptos, enquanto indivíduos menos aptos tendem a desaparecer. Eles empregam uma estratégia de busca paralela e estruturada, mas aleatória, que é voltada em direção ao reforço da busca de pontos de "alta aptidão", ou seja, pontos nos quais a função a ser minimizada (ou maximizada) tem valores relativamente baixos (ou altos).

Apesar de aleatórios, eles não são explorações aleatórias não direcionadas, pois exploram informações históricas para encontrar novos pontos de busca onde são esperados melhores desempenhos. Isto é feito através de processos iterativos, onde cada iteração é chamada de geração.

Nos AG's cada cromossomo (indivíduos da população) está associado a uma solução e cada gene está associado a um componente da solução. Durante cada iteração (geração), os princípios de seleção e reprodução são aplicados a uma população de candidatos que pode variar, dependendo da complexidade do problema e dos recursos computacionais disponíveis. Através da seleção, são determinados quais indivíduos conseguirão se reproduzir, gerando um número determinado de descendentes para a próxima geração, com uma probabilidade determinada pelo seu índice de aptidão. Em outras palavras, os indivíduos com maior adaptação têm maiores chances de se reproduzir.

Atualmente outras técnicas de busca local têm sido aplicadas de forma conjunta aos AG's no sentido de melhorar seu desempenho final. A aplicação de busca local nos indivíduos pode ser relacionada com a combinação de aprendizado e evolução (efeito Baldwin, Whitley *et al.* (1994)). O aprendizado é, em geral, uma busca local pela solução viável mais próxima e as modificações ocorridas serão incorporadas pelo indivíduo. Portanto, o que vêm acontecendo recentemente é o uso dos AG's através da conservação de suas características evolutivas associada à utilização de heurísticas específicas para prover melhoramentos individuais nos cromossomos.

4.3.3 Método de Pesquisa em Vizinhança Variável

O Método de Pesquisa em Vizinhança Variável (do inglês *Variable Neighborhood Search*, VNS), proposto por Mladenovic e Hansen (1997), é um método de busca local que realiza uma exploração cada vez mais distante da solução corrente, ao contrário dos outros métodos de busca local que seguem determinadas trajetórias de busca. Isto porque sua busca consiste em explorar o espaço de soluções através de trocas sistemáticas nas estruturas de vizinhança, focalizando a busca em torno de uma nova solução se e somente se esta apresentar alguma melhora em relação à solução corrente. O método possui também uma etapa onde um procedimento de busca local deve ser aplicado sobre a solução corrente.

Esta busca local também pode fazer uso de diferentes estruturas de vizinhança, tal como o próprio VNS.

Assim como outros métodos de busca local, o VNS parte de uma solução inicial qualquer e a cada iteração realiza um movimento dentro de uma das $N^{(k)}(s)$ estruturas de vizinhança, gerando uma nova solução s'. Essa nova solução s' é então submetida ao procedimento de busca local, gerando uma solução s''. Se a solução s'' for melhor que a solução s' então ela é aceita como nova solução corrente s e o procedimento retorna à primeira estrutura de vizinhança $N^{(1)}(s)$.

O procedimento acima é repetido até que o critério de parada seja atingido. Os critérios de parada geralmente utilizados no VNS são: tempo máximo de processamento, número máximo de iterações e número máximo de iterações consecutivas sem melhora. A Figura 4.5 apresenta o pseudocódigo do VNS para um problema de minimização.

```
Procedimento VNS(f(.), N(.), r)
1  Seja s_0 uma solução inicial;
2  Seja nv o número de estruturas diferentes de vizinhança;
3  s ← s_0; { Solução corrente }
4  enquanto (Critério de parada não for satisfeito) faça
5      k ← 1;
6      enquanto (k ≤ nv) faça
7          Gere um vizinho qualquer s' ∈ N_(k)(s);
8          s'' ← BuscaLocal(s');
9          se (f(s'') < f(s)) então
10             s ← s'';
11             k ← 1;
12         senão
13             k ← k + 1;
14         fim-se;
15     fim-enquanto;
16 fim-enquanto;
17 Retorne s;
fim-VNS;
```

Figura 4.5 - Algoritmo *VNS*

Fonte: Souza (2005)

4.3.4 *Iterated Local Search*

O método *Iterated Local Search* (ILS), ou Busca Local Iterativa, foi proposto por Lourenço *et al.* (2002). É um método que procura focar sua busca num subespaço definido por soluções que são ótimas locais de um determinado mecanismo de otimização. O sucesso do ILS é centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha da busca local, da perturbação aplicada sobre a solução corrente e o critério de aceitação das soluções.

Seu funcionamento, basicamente, segue os seguintes passos. Seja S o conjunto de todas as soluções possíveis e s uma solução candidata. A idéia é explorar S^*, que representa um conjunto menor de S somente com soluções s^*, localizadas em ótimos locais. Para isso, usa-se um caminho que leva uma solução s^* para uma próxima solução, independente desta solução ser o vizinho mais próximo. A partir da solução corrente s^*, o método aplica uma perturbação que faz determinadas mudanças nesta solução, a fim de levá-la a um estado intermediário s', também pertencente a S. A próxima etapa é a aplicação de uma busca local em s', levando-a a um ótimo local s'^* em S^*. Se s'^* atender o critério de aceitação, ela passa a ser a próxima solução na caminhada em S^*, senão retorna-se a s^*. A Figura 4.6 apresenta o pseudocódigo do ILS.

```
Procedimento ILS(.)
1  s' ← GeraSolucaoInicial();
2  s ← BuscaLocal(s');
3  enquanto (os critérios de parada não estiverem satisfeitos) faça
4      s' ← Perturbacao(histórico; s);
5      s'' ←BuscaLocal(s');
6      s ← CriterioAceitacao(s; s''; histórico);
7  fim-enquanto;
fim-ILS;
```

Figura 4.6 - Algoritmo *ILS*

Fonte: Souza (2005)

Esse mecanismo pode encontrar regiões promissoras no espaço de soluções. Para que isso seja possível, a perturbação aplicada nas soluções candidatas não pode gerar alterações muito pequenas nem alterações muito grandes. Se as alterações resultantes de uma perturbação forem muito pequenas, a nova solução s' pertencerá, freqüentemente, à região de s^*, e com isso novas soluções de S^* serão exploradas. Se a perturbação resultar em grandes alterações, s' pertencerá a uma outra região, aleatória, requerendo o reinício do

algoritmo.

O potencial do ILS está diretamente ligado às amostragens do conjunto de soluções localizadas em ótimos locais. A eficiência dessa amostragem depende tanto dos tipos de perturbações quanto do critério de aceitação.

4.3.5 *Evolutionary Clustering Search*

Trata-se de uma técnica desenvolvida recentemente na tese de Oliveira (2004) e que deriva dos Algoritmos Evolutivos. O *Evolutionary Clustering Search* (ECS), ou busca evolutiva através de agrupamentos, consiste num processo de agrupamento de indivíduos para guiar uma busca evolutiva, sendo usado para gerar um conjunto de soluções de referência para regiões, supostamente, promissoras.

A idéia de detectar regiões promissoras pode ser justificada pelo fato de possibilitar a aplicação de um algoritmo de busca local somente em indivíduos realmente promissores, diminuindo assim o número de avaliações de solução, ou seja, de chamadas à função objetivo que, para problemas reais podem ter um custo computacional muito elevado.

No *Evolutionary Clustering Search*, um processo de agrupamento iterativo trabalha simultaneamente com um algoritmo evolutivo, contabilizando as operações ocorridas em regiões do espaço de busca e identificando aquelas que merecem interesse especial. Em outras palavras, o processo de agrupamento identifica grupos de indivíduos com similaridades, significando um padrão predominante de genótipos e conseqüentemente uma certa região do espaço de busca. Espera-se uma melhoria no processo de convergência associado a uma diminuição no esforço computacional em virtude do emprego mais racional da busca local.

O ECS visa localizar regiões promissoras através do enquadramento destas em *clusters*. Um *cluster* é definido pela tripla $G = \{c; r; b\}$ onde c e r são, respectivamente, o centro e o raio de uma área de busca promissora. Existe também uma estratégia de busca b associada ao *cluster*.

O centro é um indivíduo (ou solução), representante do *cluster*, que identifica a sua localização dentro do espaço de busca. O raio estabelece a distância máxima, a partir do centro, até a qual um indivíduo pode ser associado ao *cluster*.

Em um problema de otimização combinatória, o raio pode ser definido em termos de número de movimentos necessários para transformar uma solução candidata em outra dentro de uma vizinhança definida por uma heurística qualquer.

A estratégia b é uma sistemática de intensificação de busca, na qual indivíduos de um cluster interagem entre si, ao longo do processo de agrupamento, gerando novos indivíduos na mesma região.

O ECS consiste em quatro componentes com atribuições diferentes e conceitualmente independentes. São eles: um algoritmo evolutivo (AE); um agrupador iterativo (AI); um analisador de agrupamentos (AA); e um algoritmo de otimização local (AO). A Figura 4.7 ilustra o diagrama conceitual do ECS.

O algoritmo evolutivo (AE) trabalha como um gerador de soluções de tempo integral. A população evolui independentemente dos componentes restantes. Indivíduos são selecionados, recombinados, e atualizados para as próximas gerações. Simultaneamente, *clusters* são mantidos para representar estes indivíduos.

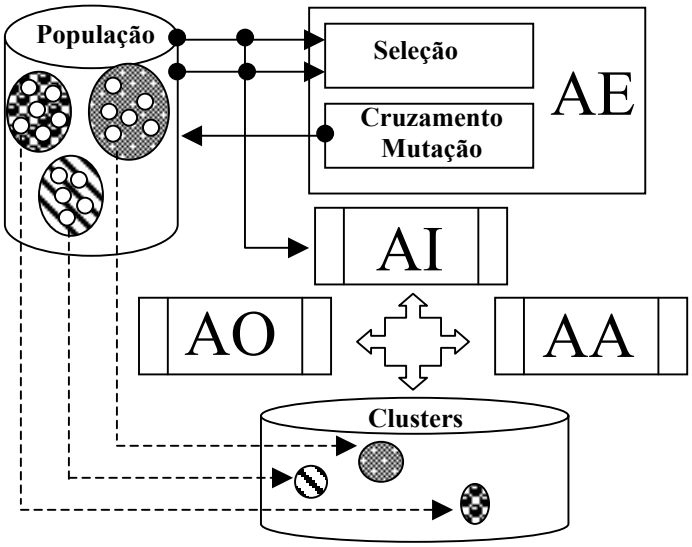

Figura 4.7 - Diagrama Conceitual do ECS.

Fonte: Oliveira (2004)

O agrupador iterativo (AI) é o núcleo do ECS, trabalhando como um classificador de informação que mantém no sistema apenas aquela que for relevante para o processo de intensificação de busca. O termo informação é usado aqui porque os indivíduos não se agrupam diretamente, mas a informação semelhante que eles representam. Toda informação gerada por AE é lida por AI que tenta agrupá-la como uma informação conhecida. Se a informação for considerada suficientemente nova, ela é mantida como um centro em um novo *cluster*. Caso contrário, a informação é considerada redundante,

causando perturbação no centro do *cluster* mais similar. Tal perturbação é chamada de assimilação e consiste basicamente em atualizar o centro com a nova informação recebida.

O analisador de agrupamentos (AA) provê uma análise de cada agrupamento (*cluster*), em intervalos regulares de gerações, indicando um provável grupo promissor. Tipicamente, a densidade do *cluster* é usada nesta análise, ou seja, o número de seleções ou atualizações que aconteceram recentemente no *cluster*. Um *cluster* com densidade alta possui, provavelmente, um centro promissor. O AA também é responsável pela eliminação de *clusters* com baixas densidades.

Por fim, o algoritmo de otimização local (AO) é um módulo de pesquisa interno que provê a exploração de uma suposta região promissora. Este processo acontece depois que o componente AA tenha descoberto um *cluster* alvo. A busca local é realizada no indivíduo que está no centro do *cluster*.

5 METODOLOGIA DE TRABALHO

Neste trabalho são propostas trê abordagens, utilizando algoritmos híbridos, aplicados na resolução do *Mirrored Traveling Tournament Problem* - mTTP (ver Seção 2.2). A escolha desta variante do TTP se deve ao fato deste problema representar uma modalidade esportiva bastante comum em competições realizadas na América Latina, portanto mais próxima da nossa realidade. Além de possuir instâncias públicas que vêm recebendo, recentemente, muita atenção da comunidade científica.

A primeira abordagem trata o problema a partir do uso conjunto de técnicas evolutivas e busca local. Foi implementado um Algoritmo Genético híbrido, que faz uso da técnica *Simulated Annealing* para conduzir cada filho gerado a um ótimo local. Neste trabalho, esta abordagem híbrida será referenciada como *GA-SA*.

Para utilização do *GA-SA* é proposta uma nova forma de se representar o problema, através de uma representação compacta de cromossomos, associada a um algoritmo de expansão de código que faz a decodificação destes cromossomos em escalas de jogos.

A segunda abordagem é uma adaptação da primeira, pois foi adicionado ao *GA-SA* um algoritmo que realiza buscas através de agrupamento de soluções. Esta técnica é conhecida como *Evolutionary Clustering Search* (ECS) e foi proposta por Oliveira (2004). Basicamente, a idéia é utilizar o *GA-SA* como componente evolutivo do ECS (ver Subseção 4.3.5).

Na terceira abordagem será considerada uma adaptação do ECS, conhecida como *CS, onde outras metaheurísticas são utilizadas de forma alternativa ao algoritmo evolutivo, na etapa de geração de soluções para o processo de agrupamento. Um exemplo de utilização do *CS pode ser encontrado em Oliveira e Lorena (2006b). Neste trabalho, será utilizada a técnica *Variable Neighborhood Search* (VNS), gerando a abordagem *VNS-CS*. Tal abordagem foi empregada com sucesso em Chaves e Lorena (2005).

5.1 Modelagem do Problema

5.1.1 Representação de uma Solução

Uma solução, escala de jogos s, é representada por uma tabela indicando os oponentes de cada time em cada uma das rodadas. Cada linha da tabela corresponde a um time e cada coluna corresponde a uma rodada de jogos (ver Figura 5.1) . O oponente do time T_i na rodada r_k é representado pelo valor absoluto do elemento (i, k). Se (i, k) é positivo, o jogo é realizado na sede do time T_i, caso contrário na sede do time T_j. O número de

rodadas representadas é igual a $(n-1)$, onde n é o número de times. Este número de rodadas equivale apenas aos jogos do turno. Optou-se por não representar o returno pelo fato dos jogos deste terem a mesma disposição dos jogos do turno, porém com os mandos de campos invertidos (espelho do turno).

Considerando um torneio com 6 times (portanto 5 rodadas no turno e 5 rodadas espelhadas no returno), uma possível escala para este torneio é apresentada abaixo.

	Turno					**Returno**				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	-3	-4	5	-6	-2	3	4	-5	6	2
2	6	5	3	-4	1	-6	-5	-3	4	-1
3	1	6	-2	5	4	-1	-6	2	-5	-4
4	-5	1	6	2	-3	5	-1	-6	-2	3
5	4	-2	-1	-3	6	-4	2	1	3	-6
6	-2	-3	-4	1	-5	2	3	4	-1	5

Figura 5.1 - Representação de uma solução.

5.2 Estruturas de Vizinhança

A vizinhança de uma escala s é o conjunto de escalas que podem ser obtidas pela aplicação de um dos cinco tipos de movimentos, descritos abaixo, sempre aplicados na escala referente ao turno, mas refletindo, conseqüentemente, na escala do returno.

5.2.1 Troca Mando de Campo (TMC)

O movimento Troca Mando de Campo (TMC) realiza a troca do mando de campo de um jogo envolvendo dois times, T_i e T_j, sendo T_i escolhido aleatoriamente e T_j o seu oponente, em uma rodada r_k também escolhida aleatoriamente. Em outras palavras, se o time T_i joga com o time T_j na cidade sede de T_j, na rodada r_k, o movimento de trocar o mando de campo faz com que o time T_i passe a jogar em sua cidade sede, conseqüentemente o time T_j deve se deslocar até a sede de T_i, na rodada r_k . Considere a escala S apresentada na Figura 5.2, bem como o time T_1 e a rodada r_2.

O movimento Troca Mando de Campo aplicado no time T_1, rodada r_2, gera a escala s', apresentada na Figura 5.3.

	Turno					Returno				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	-3	**-4**	5	-6	-2	3	**4**	-5	6	2
2	6	5	3	-4	1	-6	-5	-3	4	-1
3	1	6	-2	5	4	-1	-6	2	-5	-4
4	-5	**1**	6	2	-3	5	**-1**	-6	-2	3
5	4	-2	-1	-3	6	-4	2	1	3	-6
6	-2	-3	-4	1	-5	2	3	4	-1	5

Figura 5.2 - O time T_1 é visitante no jogo contra o time T_4, na rodada r_2

	Turno					Returno				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	-3	**4**	5	-6	-2	3	**-4**	-5	6	2
2	6	5	3	-4	1	-6	-5	-3	4	-1
3	1	6	-2	5	4	-1	-6	2	-5	-4
4	-5	**-1**	6	2	-3	5	**1**	-6	-2	3
5	4	-2	-1	-3	6	-4	2	1	3	-6
6	-2	-3	-4	1	-5	2	3	4	-1	5

Figura 5.3 - O time T_1 é mandante no jogo contra o time T_4, na rodada r_2

5.2.2 Troca Times (TT)

O movimento Troca Times (TT) permuta os oponentes de dois times, T_i e T_j, escolhidos aleatoriamente. Apenas os jogos onde os times escolhidos se confrontam não serão alterados (Figura 5.4). Em contrapartida, outros dois oponentes de cada um dos outros times também serão trocados, ou seja, os times que jogavam contra T_i passam a jogar com T_j e vice-versa.

Outra questão importante neste movimento é que, dado que T_i ficará com a seqüência de jogos de T_j e T_j com a seqüência de jogos de T_i, a sede do jogo entre estes dois times tem que ser invertida, para que a viabilidade da tabela gerada seja mantida.

	Turno					Returno				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	-3	-4	5	-6	-2	3	4	-5	6	2
2	6	5	3	-4	1	-6	-5	-3	4	-1
3	**1**	**6**	**-2**	5	**4**	**-1**	**-6**	**2**	-5	**-4**
4	-5	1	6	2	-3	5	-1	-6	-2	3
5	**4**	**-2**	**-1**	-3	**6**	**-4**	**2**	**1**	3	**-6**
6	-2	-3	-4	1	-5	2	3	4	-1	5

Figura 5.4 - Escala s antes da aplicação do movimento TT.

A aplicação do movimento Troca Times em T_3 e T_5 gera uma escala s', apresentada na Figura 5.5.

	Turno					Returno				
T_i/r_k	1	2	3	4	5	6	7	8	9	10
1	**-5**	-4	**3**	-6	-2	**5**	4	**-3**	6	2
2	6	**3**	**5**	-4	1	-6	**-3**	**-5**	4	-1
3	**4**	**-2**	**-1**	-5	**6**	**-4**	**2**	**1**	5	**-6**
4	**-3**	1	6	2	**-5**	**3**	-1	-6	-2	**5**
5	**1**	**6**	**-2**	3	**4**	**-1**	**-6**	**2**	-3	**-4**
6	-2	**-5**	-4	1	**-3**	2	**5**	4	-1	**3**

Figura 5.5 - Escala s' gerada pela aplicação do movimento TT em T_3, T_5.

5.2.3 Troca Rodadas (TR)

Dadas duas rodadas, r_i e r_j, aleatoriamente escolhidas, o movimento Troca Rodadas (TR) permuta todos os oponentes de todos os times nas duas rodadas. Considere as rodadas r_2 e r_4 da escala S apresentada na Figura 5.6:

	Turno					Returno				
T_i/r_k	1	2	3	4	5	6	7	8	9	10
1	-3	**-4**	5	**-6**	-2	3	**4**	-5	**6**	2
2	6	**5**	3	**-4**	1	-6	**-5**	-3	**4**	-1
3	1	**6**	-2	**5**	4	-1	**-6**	2	**-5**	-4
4	-5	**1**	6	**2**	-3	5	**-1**	-6	**-2**	3
5	4	**-2**	-1	**-3**	6	-4	**2**	1	**3**	-6
6	-2	**-3**	-4	**1**	-5	2	**3**	4	**-1**	5

Figura 5.6 - Escala s antes da aplicação do movimento TR.

A aplicação do movimento Troca Rodadas em r_2 e r_4 gera a escala s' apresentada na Figura 5.7.

	Turno					Returno				
T_i/r_k	1	2	3	4	5	6	7	8	9	10
1	-3	**-6**	5	**-4**	-2	3	**6**	-5	**4**	2
2	6	**-4**	3	**5**	1	-6	**4**	-3	**-5**	-1
3	1	**5**	-2	**6**	4	-1	**-5**	2	**-6**	-4
4	-5	**2**	6	**1**	-3	5	**-2**	-6	**-1**	3
5	4	**-3**	-1	**-2**	6	-4	**3**	1	**2**	-6
6	-2	**1**	-4	**-3**	-5	2	**-1**	4	**3**	5

Figura 5.7 - Escala s' gerada pela aplicação do movimento TR em r_2, r_4.

5.2.4 Troca Parcial de Rodadas (TPR)

Sejam os times T_i, T_j, T_k e T_l, e as rodadas r_i e r_j nas quais os jogos $\{T_i, T_k\}$ e $\{T_j, T_l\}$ acontecem na rodada r_i e os jogos $\{T_i, T_l\}$ e $\{T_j, T_k\}$ na rodada r_j. O movimento Troca Parcial de Rodadas consiste em trocar os jogos que envolvem estes times nas respectivas rodadas, r_i e r_j, ou seja, os jogos $\{T_i, T_k\}$ e $\{T_j, T_l\}$ passam para a rodada r_j e os jogos $\{T_i, T_l\}$ e $\{T_j, T_k\}$ passam para a rodada r_i.

Seja a escala s apresentada na Figura 5.8, os times T_1, T_2, T_4 e T_6 e as rodadas r_1 e r_3.

	Turno					Returno				
T_i/r_k	1	2	3	4	5	6	7	8	9	10
1	-2		-6			2		6		
2	1		4			-1		-4		
3										
4	-6		-2			6		2		
5										
6	4		1			-4		-1		

Figura 5.8 - Fragmento de uma escala s antes da aplicação do movimento TPR.

A aplicação do movimento TPR considerando os jogos $\{T_1, T_2\}$ e $\{T_4, T_6\}$ programados na rodada r_1 e os jogos $\{T_1, T_6\}$ e $\{T_4, T_2\}$ na rodada r_3 gera a escala s' apresentada na Figura 5.9.

	Turno					Returno				
T_i/r_k	1	2	3	4	5	6	7	8	9	10
1	-6		-2			6		2		
2	4		1			-4		-1		
3										
4	-2		-6			2		6		
5										
6	1		4			-1		-4		

Figura 5.9 - Fragmento da escala s' gerada pela aplicação do movimento TPR.

É importante salientar que o movimento TPR não é sempre possível de se realizar devido ao molde imposto às soluções geradas pelo Método do Polígono (descrito na Seção 5.4), como por exemplo, para instâncias onde $n = 6$, 8, 12, 14, 16, 20 e 24, sendo n o número de times. Porém, a possibilidade de se realizar o movimento pode aparecer após a execução do movimento Troca Jogos (descrito na Subseção 5.2.5) em instâncias onde $n \geq 8$.

5.2.5 Troca Jogos (TJ)

O movimento Troca Jogos consiste em programar um jogo $\{T_i, T_j\}$ em uma rodada r_k aleatoriamente escolhida. Conseqüentemente, outros jogos deverão ser programados em outras rodadas para que seja mantida a viabilidade da escala, o que gera uma seqüência de trocas de jogos. Tais modificações que devem ser aplicadas à solução corrente geram um movimento conhecido na literatura como *ejection chain* (GLOVER, 1992; GLOVER, 1996a), bastante usado em problemas de otimização que envolvem rotas.

Seja s a escala apresentada na Figura 5.10.

	Turno					Returno				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	-6	4	2	-5	-3	6	-4	-2	5	3
2	-5	3	-1	-4	-6	5	-3	1	4	6
3	4	-2	-5	6	1	-4	2	5	-6	-1
4	-3	-1	6	2	-5	3	1	-6	-2	5
5	2	-6	3	1	4	-2	6	-3	-1	-4
6	1	5	-4	-3	2	-1	-5	4	3	-2

Figura 5.10 - Escala s antes da aplicação do movimento TJ.

A aplicação do movimento Troca Jogos na escala s, programando-se o jogo $\{T_1, T_3\}$ na rodada r_2 (programado em r_5 na escala s), gera uma escala s' onde vários jogos foram trocados, conforme apresentado na Figura 5.11.

	Turno					Returno				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**-5**	**-3**	2	**4**	**-6**	5	3	-2	-4	6
2	**-6**	**4**	-1	**-5**	**3**	6	-4	1	5	-3
3	-4	**1**	-5	**6**	**-2**	4	-1	5	-6	2
4	3	**-2**	6	**-1**	**5**	-3	2	-6	1	-5
5	**1**	-6	3	**2**	**-4**	-1	6	-3	-2	4
6	**2**	5	-4	-3	**1**	-2	-5	4	3	-1

Figura 5.11 - Escala s' gerada pela aplicação do movimento TJ.

Após a aplicação deste movimento, uma nova configuração dos mandos de campo deve ser realizada caso os mandos originados pelo movimento gerem uma solução inviável. Para tal, o método de busca local RNA foi aplicado na recuperação da viabilidade.

O movimento Troca Jogos é muito importante na busca por boas soluções para o mTTP. Isso porque, este movimento é capaz de encontrar soluções que não podem ser alcançadas pelas outras estruturas de vizinhança.

5.3 Função de Avaliação

Problemas de geração de escalas de jogos não possuem uma função de avaliação tão natural quanto as funções de muitos outros problemas de otimização. As escalas de jogos das competições programadas manualmente, em geral, exprimem o resultado de uma negociação entre os clubes participantes, a entidade organizadora e as redes de televisão, iteradas por várias tentativas de conciliar interesses legítimos, mas muitas vezes mutuamente excludentes. É, portanto, uma tarefa bastante complexa construir uma função de avaliação que exprima de forma fiel a interação entre os inúmeros aspectos a considerar. A solução que será adotada neste trabalho é inspirada no trabalho de Costa (1995), que avalia uma escala s através da Equação 5.1 apresentada a seguir.

$$F(s) = \sum_{i \in T} D_i + \sum_{j \in C} (w_j * f_j(s)) \tag{5.1}$$

Na função acima os componentes são descritos individualmente da seguinte forma:

- $F(s)$: 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 de índice i pertencente ao conjunto de times T;
- w_j: peso que representa a importância relativa de se atender a restrição de índice j pertencente ao conjunto de restrições C e;
- $f_j(s)$: função que computa o grau de violação da j-ésima restrição pertencente ao conjunto C.

Os valores dos pesos devem ser escolhidos de forma a se produzir soluções comparáveis com as produzidas manualmente, bem como os resultados apresentados na literatura. Neste trabalho foi utilizado o valor 10^9 para todo peso w_i.

Sendo assim, a função de avaliação considerada neste trabalho, que deve ser minimizada, é baseada em penalidade e composta por algumas partes distintas, que representam as restrições descritas na Seção 2.2.

5.4 Geração de Solução Inicial

A geração de solução inicial é uma etapa de suma importância para problemas de otimização. Boas soluções iniciais podem afetar, de forma positiva, a performance dos algoritmos (geralmente metaheurísticas) destinados a esta tarefa, como o caso do *Simulated Annealing*. No trabalho desenvolvido por Elmohamed *et al.* (1998) é apresentada uma análise de desempenho do SA quando submetido a técnicas diferentes de geração de solução inicial. Os resultados demonstraram que para soluções iniciais geradas de forma aleatória, a performance da metaheurística *Simulated Annealing* não é muito boa, necessitando de muito processamento computacional e tempo para encontrar soluções de boa qualidade. Por outro lado, quando o ponto de partida do algoritmo é uma solução de qualidade "razoável", há uma melhora significativa na performance da metaheurística.

Por estes e outros motivos o uso de algoritmos aproximativos rápidos é muito importante em problemas relacionados a escalonamento esportivo, como o caso do TTP, uma vez que estes apresentam uma dependência muito grande entre suas restrições.

Neste trabalho, o Método do Polígono (DINITZ *et al.*, 1995) é utilizado na geração da escala de jogos do primeiro turno de um mTTP. O segundo turno, ou returno, é o espelho do primeiro turno, portanto não é necessária a sua representação (ver Figura 5.1). Associado a estes métodos é utilizado um algoritmo que alterna os mandos de campo dos jogos disputados por cada time, inspirado no trabalho de Ribeiro e Urrutia (2004). Por fim, uma heurística de busca local é aplicada à escala para conduzir as soluções geradas a ótimos locais. A seguir, cada um desses passos é descrito em detalhes, começando pela aplicação do Método do Polígono.

Considere um vetor V de tamanho n (número de times) onde cada posição $V[i]$ representa um time. A execução do método começa com a escolha aleatória de um time base, que é inserido na primeira posição do vetor V. Os demais times são dispostos nas posições restantes ($i = 2, ..., n$). Em cada rodada $r_k = 1, ..., n-1$ o time base (posição $i = 1$) confronta o time da posição $i = 2$, definindo o jogo $\{T_{V[1]}, T_{V[2]}\}$. Os times das posições $i = 3, ..., (n/2) + 1$ confrontam os times das posições $j = n - i + 3$, definindo os demais jogos $\{T_{V[i]}, T_{V[j]}\}$. Uma vez definidos todos os jogos da rodada r_k, os times das posições $i = 2, ..., n$ são rotacionados no sentido horário do vetor, da seguinte forma: os times das posições $i = 3, ..., n$ passam para a posição $i-1$ e o time da posição 2 passa para a posição n. A Figura 5.12 apresenta a aplicação do Método do Polígono, dado um vetor de times, onde $n = 6$, sendo 3 o time base.

Rodada 1	**3**	2	5	1	6	4
Rodada 2	**3**	5	1	6	4	2
Rodada 3	**3**	1	6	4	2	5
Rodada 4	**3**	6	4	2	5	1
Rodada 5	**3**	4	2	5	1	6

Figura 5.12 - Geração das rodadas para n = 6, usando vetores de times.

A Figura 5.13 ilustra o método aplicado em polígonos.

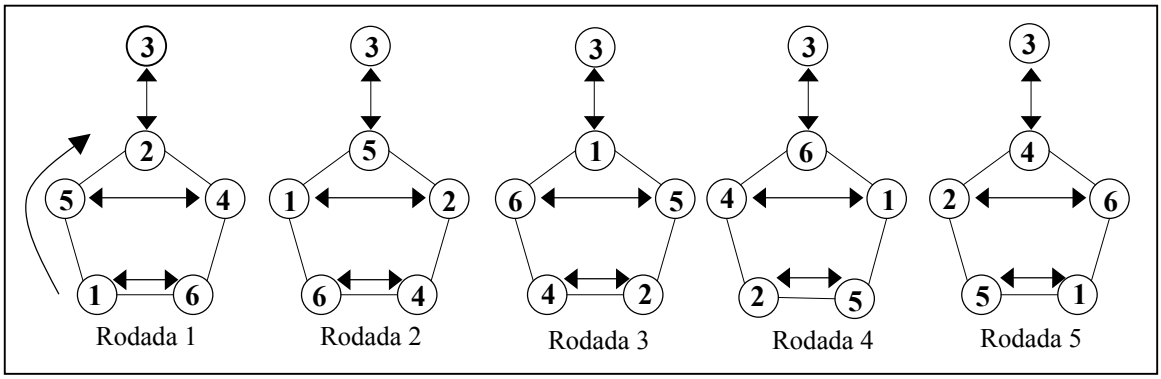

Figura 5.13 - Ilustração de uma geração de rodadas para n = 6, usando polígonos.

Uma vez definida a escala de jogos, a próxima etapa é a atribuição de mandos de campo a cada um dos jogos da escala. Fazer com que cada time dispute o maior número possível de jogos dentro ou fora de sua sede é uma boa estratégia para minimizar as distâncias de viagem dos times, uma vez que eles não necessitariam retornar para sua sede após os jogos. Baseado neste fato, foi utilizado um algoritmo que alterna os mandos de campo dos jogos disputados por cada time, sendo este descrito a seguir.

Define-se mandos de campo aleatórios para os jogos da primeira rodada. Da segunda rodada em diante, a seguinte estratégia é utilizada para atribuir mandos de campo aos jogos definidos pelos times $\{T_i, T_j\}$. Considere N_i (respectivamente N_j) o número de jogos que o time T_i (respectivamente T_j) jogou consecutivamente, tanto dentro ou fora de casa, nas rodadas anteriores.

- Se $N_j > N_i$ então
  - Se T_j foi mandante em seu último jogo, então T_i deve ser mandante e T_j o visitante no jogo corrente;

- Caso contrário, T_i deve ser visitante e T_j o mandante.

- Se $N_j < N_i$ então
  - Se T_i foi mandante em seu último jogo, então T_j deve ser mandante e T_i o visitante no jogo corrente;
  - Caso contrário, T_j deve ser visitante e T_i o mandante.

- Se $N_j = N_i$ então
  - Se T_i foi mandante e T_j visitante em seus últimos jogos, então T_i deve ser visitante e T_j o mandante no jogo corrente;
  - Se T_j foi mandante e T_i visitante em seus últimos jogos, então T_j deve ser visitante e T_i o mandante no jogo corrente;
  - Caso contrário, sortear o time mandante.

Esta estratégia pode resultar ao final da atribuição dos mandos de campo em uma escala inviável. A primeira rodada do segundo turno é composta pelos mesmos jogos da primeira rodada do primeiro turno, porém com os mandos de campo invertidos. Em conseqüência, a última rodada do primeiro turno e a primeira do segundo turno devem ser consideradas como rodadas consecutivas e isso pode gerar algumas inviabilidades. Resultados computacionais obtidos por Ribeiro e Urrutia (2004) mostraram que esta situação acontece apenas em cerca de $6,3\%$ das vezes em que a estratégia é aplicada nas instâncias de teste, sendo que para nenhuma delas esse valor supera 8%.

Para que a solução inicial gerada seja uma escala viável e de boa qualidade, o método RNA, realizando apenas o movimento Troca Mando de Campo, é aplicado à escala, para tentar eliminar possíveis inviabilidades, bem como minimizar as distâncias percorridas pelos times, até que um ótimo local seja atingido.

A partir desses passos é possível construir soluções iniciais viáveis de forma rápida, deixando o esforço computacional concentrado na avaliação das vizinhanças das mesmas. Um ponto negativo desta abordagem é que toda solução gerada pela estratégia se encontra dentro de um molde, imposto pelo Método do Polígono. Este molde só poderá ser alterado através da aplicação dos movimentos Troca Parcial de Rodadas (ver Subseção 5.2.4) e Troca Jogos (ver Subseção 5.2.5).

5.5 *GA-SA* Aplicado ao mTTP

A aplicação de algoritmos evolutivos na resolução de problemas com restrições que devem ser atendidas é interessante sob dois aspectos, segundo Eiben e Ruttkay (1997). O primeiro aspecto a ser levado em consideração é que não se pode garantir que um algoritmo clássico de busca local possa alcançar com facilidade a solução de um problema com restrições. Determinadas heurísticas se mostram bastante eficientes na resolução de determinados problemas, mas falham em outros por restringir o espaço de busca. O segundo aspecto é que algumas instâncias de problemas com restrições podem ser resolvidas através de uma busca mais diversificada, pela manutenção de vários candidatos à solução em paralelo e pela aplicação de heurísticas que agreguem mecanismos de construção aleatória de novos candidatos à solução. Como essas características são o ponto forte dos algoritmos evolutivos, a sua aplicação na resolução de problemas com restrições apresenta-se como uma alternativa promissora.

Por outro lado, a utilização de algoritmos evolutivos é tradicionalmente realizada em problemas sem restrições. Como os operadores tradicionais de mutação e recombinação não tratam diretamente restrições, a aplicação de estratégias evolutivas na resolução de problemas com estas características não é realizada de forma imediata, além de não garantir que "pais" que representem soluções viáveis gerem "filhos" também viáveis.

Considerando estes fatos é proposta, neste trabalho, a aplicação conjunta de algoritmos evolutivos e busca local. Utilizou-se uma abordagem híbrida, associando os Algoritmos Genéticos (ver Subseção 4.3.2) com o método *Simulated Annealing* (ver Subseção 4.3.1), sendo o objetivo deste último, o direcionamento dos novos indivíduos de cada geração a ótimos locais. A idéia da aplicação de busca local nos indivíduos pode ser relacionada com a combinação de aprendizado e evolução (efeito *Baldwin*, Whitley *et al.* (1994)). O aprendizado é, em geral, uma busca local pela solução viável mais próxima e as modificações ocorridas serão incorporadas pelo indivíduo.

Embora na literatura haja outras abordagens evolutivas aplicadas ao TTP, a abordagem proposta neste trabalho inova ao sugerir a utilização de uma representação genética compacta (cromossomos) associada a um algoritmo de expansão de código, que tem por função a decodificação dos cromossomos em escalas de jogos (BIAJOLI; LORENA, 2005; BIAJOLI; LORENA, 2006). Desta forma, o AG não trabalha diretamente com escalas de jogos, pois esta tarefa é a grande, senão a maior, dificuldade de aplicar algoritmos evolutivos na resolução do TTP.

Assim sendo, a aplicação conjunta das metodologias citadas se justifica por dois pontos

principais:

- O processo evolutivo vai poder atuar somente sobre a codificação compacta, junto a qual não há inviabilidade, fazendo com que sua eficácia não seja reduzida;
- A existência de uma população de candidatos à solução sendo evoluída, juntamente com a aplicação de um procedimento de busca local, aumenta significativamente as chances de se obter ótimos locais de boa qualidade.

Um cromossomo é representado por um vetor onde cada posição (gene) está associado a um time. A Figura 5.14 apresenta um exemplo de cromossomo, para um torneio com 6 times participantes, logo, 6 genes.

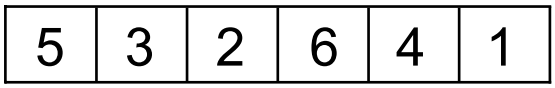

Figura 5.14 - Exemplo de um cromossomo para um torneio com 6 times.

Um mecanismo de reprodução, baseado em processos evolutivos, é aplicado sobre a população com o objetivo de explorar o espaço de busca e encontrar melhores soluções para o problema. O operador de recombinação utilizado para o cruzamento é o *Block Order Crossover* (BOX, Syswerda (1989)). Os indivíduos pais são combinados através da cópia aleatória de blocos de ambos indivíduos, o que resulta na geração de um único filho, contendo carga genética dos dois pais. Blocos de genes copiados de um indivíduo não são copiados do outro, mantendo a viabilidade da solução. A Figura 5.15 ilustra o operador de recombinação BOX.

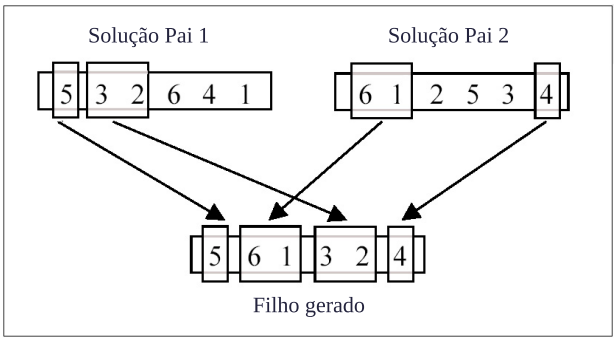

Figura 5.15 - Exemplo de recombinação usando o operador BOX

Em cada geração dois indivíduos são aleatoriamente selecionados dentro da população,

sendo um deles escolhido entre os 70% mais aptos e o outro entre os 30% menos aptos.

Cada novo indivíduo gerado pelo processo de recombinação é submetido ao algoritmo de expansão de código, apresentado na Seção 5.4, para geração da escala de jogos com $(n-1)$ rodadas, sendo n o número de times. Este processo resulta em escalas de jogos bem diferentes a cada geração, aumentando o escopo do espaço de busca. É importante ressaltar ainda, que todas as escalas geradas por esta estratégia atendem as restrições do mTTP, portanto, são soluções viáveis para o problema.

O operador de mutação empregado consiste na aplicação do movimento Troca Jogos (ver Subseção 5.2.5), com o objetivo de, possivelmente, retirar a escala de jogos originada pelo processo de recombinação do molde que é imposto à soluções geradas pelo Método do Polígono. Este operador desempenha um papel muito importante na busca por boas soluções para o mTTP, pois o movimento Troca Jogos é capaz de encontrar soluções que não podem ser alcançadas pelas outras estruturas de vizinhança utilizadas neste trabalho.

O próximo passo é a execução de uma busca local. A metaheurística *Simulated Annealing* com os movimentos Troca Mando de Campo (TMC) e Troca Times (TT) é aplicada a cada indivíduo, retornando ao final de sua execução uma solução localizada em um ótimo local. A Figura 5.16 apresenta o pseudocódigo do *GA-SA*.

```
Procedimento GA-SA(.)
1   Inicialize a população P;
2   enquanto (g < nGeracoes) faça
3       enquanto (i < nIndividuos) faça
4           pai1 ← Selecione um individuo ∈ 70% melhores de P;
5           pai2 ← Selecione um individuo ∈ 30% piores de P;
6           filho ← Recombinacao(pai1, pai2);
7           se (atende critério de mutacao) então
8               Mutacao(filho);
9           fim-se
10          filho' ← BuscaLocal(filho);
11          Avaliacao(filho');
12          Adicione filho' em P;
13          i ← i + 1;
14      fim-enquanto
15      P ← DefinaSobreviventes(P);
16      g ← g + 1;
17  fim-enquanto
fim-GA-SA
```

Figura 5.16 - Algoritmo *GA-SA*

A população inicial é composta por 300 indivíduos e 100 novos são gerados a cada uma das 10 gerações. Um mecanismo baseado na idéia de "rolera russa" foi implementado para definir quais indivíduos serão eliminados da população ao término de cada geração. Este método consiste em associar a cada indivíduo uma fatia da roleta igual à sua probabilidade de eliminação (PE_i), valor este calculado à partir do valor da função de avaliação ($f(x)$) do indivíduo. A Equação 5.2 apresenta a fórmula utilizada para o cálculo da PE.

$$PE_i = \frac{f(x)_i}{\sum_{i=1}^{n} f(x)_i} \tag{5.2}$$

onde n é o número de times participantes da competição.

De acordo com esta expressão a probabilidade de sobrevivência de um indivíduo é proporcional ao seu nível de aptidão, ou seja, quanto menor a fatia do indivíduo na roleta maior sua aptidão. A implementação computacional deste método foi feita da seguinte forma. Depois de calculada a probabilidade de eliminação (PE_i) de cada indivíduo, é atribuída a cada um deles uma fatia (F_i) na roleta proporcional a sua PE_i. Os indivíduos são esquematizados seqüencialmente na roleta, conforme ilustrado no exemplo da Figura 5.17.

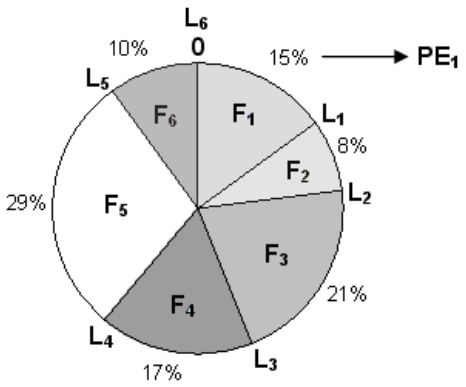

Figura 5.17 - Exemplo do mecanismo utilizado para definição da população sobrevivente

Cada indivíduo i da população tem a sua fatia (F_i) na roleta, compreendida no intervalo $L_{i-1} < F_i < L_i$ (em que $L_0 = 0$ e $L_i = L_{i-1} + PE_i$). Uma vez definido o intervalo que cada indivíduo ocupará na roleta, um número aleatório k ($0 < k < 1$) é gerado, simulando uma rodada da roleta. O indivíduo eliminado em cada rodada da roleta é aquele cujo valor de k pertence ao seu intervalo. O processo é repetido até que seja restabelecida uma população de número equivalente à população inicial.

5.6 ECS Aplicado ao mTTP

O algoritmo ECS, conforme sua descrição apresentada na Subseção 4.3.5, tem por objetivo a detecção de regiões promissoras através do agrupamento de genótipos, ou seja, similaridades entre soluções. Com isso, procura-se amenizar um dos maiores desafios dos métodos de otimização, que é a definição de estratégias eficientes para cobrir todo o espaço de busca, possibilitando a aplicação da busca local apenas nas regiões realmente promissoras.

Conforme apresentado, o ECS consiste em quatro componentes com atribuições diferentes e conceitualmente independentes. Estes componentes são detalhados nas próximas seções.

5.6.1 Componente AE

O componente AE (Algoritmo Evolutivo) utilizado neste trabalho foi o algoritmo *GA-SA*, apresentado na Seção 5.5. A idéia é utilizar este algoritmo como gerador de soluções de tempo integral. A população evoluirá independentemente dos outros componentes do ECS. Os indivíduos são selecionados, recombinados, e atualizados para as próximas gerações. Simultaneamente, é realizada a manutenção dos *clusters*, usados para representação das regiões onde os indivíduos estão localizados no espaço de busca. A cada iteração do componente AE, os dois indivíduos selecionados na população são enviados ao componente AI, que irá agrupá-los ao *cluster* mais próximo. A implementação do componente AI é apresentada na próxima seção.

5.6.2 Componente AI

O agrupador iterativo (AI) trabalha como um classificador de informação que mantém no sistema apenas aquela que for relevante para o processo de intensificação de busca. Toda informação gerada por AE é lida por AI que tenta agrupá-la como uma informação conhecida.

Neste sentido, foi utilizada uma métrica para medição da semelhança entre as soluções geradas por AE e as soluções armazenadas nos centros c_i de cada um dos NC_{max} *clusters*, onde NC_{max} é o número máximo de *clusters* que são mantidos pelo ECS. Neste trabalho, NC_{max} foi definido com o valor 20. Basicamente, a métrica utilizada consiste na diferença entre os jogos das escalas, não levando em consideração o mando de campo desses jogos (A x B é igual a B x A).

Se a solução for considerada suficientemente nova, ela é mantida como um centro em um novo *cluster*. Caso contrário, a informação é considerada redundante, causando

perturbação no centro do *cluster* mais similar. Uma solução será considerada similar a um determinado centro quando houver pelo menos 50% de coincidência entre os jogos das escalas comparadas. Portanto, o raio dos *clusters* pode ser definido da seguinte forma:

$$Raio = \left[0.5 \left(\frac{n(n-1)}{2}\right)\right] \tag{5.3}$$

onde n é o número de times participantes da competição.

A aplicação de uma perturbação no centro c_i mais similar é chamada de *Processo de Assimilação* e consiste, basicamente, em atualizar c_i com a nova informação recebida. Para esta etapa foi implementada uma assimilação por caminho, utilizando o método *Path-Relinking* (ver Subseção 4.2.3). No entanto, é realizada a assimilação para um único time, escolhido aleatoriamente. Esta opção de assimilação não evita alterações nas seqüências de jogos dos demais times, pois rodadas inteiras serão trocadas, logo, as seqüências de todos os times serão alteradas.

Considere a escala apresentada na Figura 5.18 como sendo o centro de um *cluster* c_0 e a escala da Figura 5.19 uma escala s_0, similar ao núcleo de c_0, por exemplo .

	Turno					**Returno**				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	4	-5	-3	2	-6	-4	5	3	-2	6
2	-3	4	6	-1	-5	3	-4	-6	1	5
3	2	-6	1	5	-4	-2	6	-1	-5	4
4	-1	-2	5	-6	3	1	2	-5	6	-3
5	6	1	-4	-3	2	-6	-1	4	3	-2
6	-5	3	-2	4	1	5	-3	2	-4	-1

Figura 5.18 - Escala representando o centro de um *cluster* c_0

A assimilação consiste em fazer com que as seqüências de jogos do time escolhido na escala s_0 sejam igualadas às seqüências de jogos do centro c_0. Para isso, é percorrido todo o caminho entre as duas soluções, permutando rodadas e avaliando as soluções geradas pelo caminho. A cada solução gerada pelo caminho, uma busca local que realiza apenas o movimento Troca Mando de Campo é aplicada.

Considerando que o time T_1 foi o escolhido, um exemplo do processo de assimilação é apresentado na Figura 5.20. Para cada estado da solução são realizados todas as trocas entre rodadas que levam a solução do estado corrente a um estado próximo ao centro. A

	Turno					Returno				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	-2	-3	4	5	-6	2	3	-4	-5	6
2	1	4	-6	-3	5	-1	-4	6	3	-5
3	-6	1	-5	2	4	6	-1	5	-2	-4
4	5	-2	-1	6	-3	-5	2	1	-6	3
5	-4	6	3	-1	-2	4	-6	-3	1	2
6	3	-5	2	-4	1	-3	5	-2	4	-1

Figura 5.19 - Escala S_0, similar ao centro c_0

solução de menor custo passa para o estado corrente e novas trocas são realizadas. Esse processo é realizado até que as duas seqüências de jogos sejam idênticas.

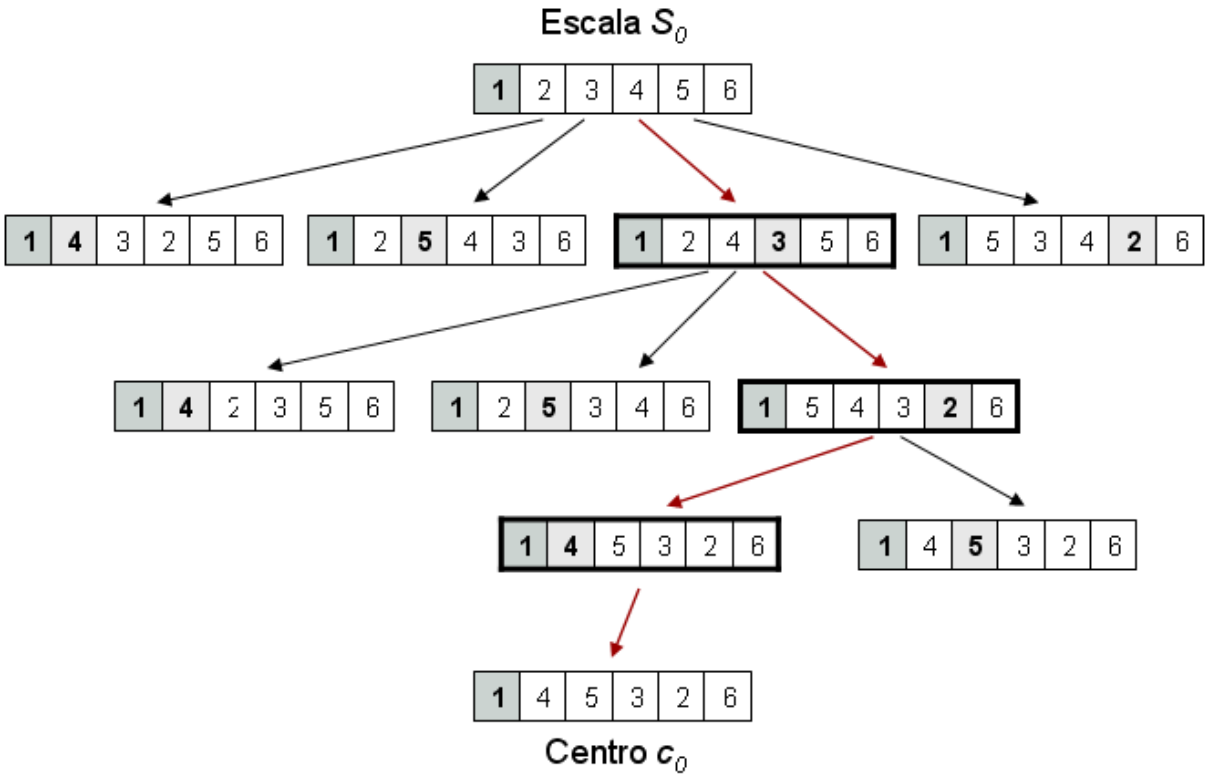

Figura 5.20 - Exemplo de assimilação aplicada no time T_1, envolvendo S_0 e c_0

O resultado do processo de assimilação é uma nova escala, sendo esta a de menor valor dentre as encontradas no caminho percorrido entre as soluções s_0 e c_0. Se considerarmos que a solução de menor custo foi a obtida pela última etapa do *Path-Relinking*, então uma possível escala de jogos é apresentada na Figura 5.21.

	Turno					Returno				
T_i/r_k	1	2	3	4	5	6	7	8	9	10
1	4	5	-3	-2	6	-4	-5	3	2	-6
2	-6	-3	4	1	-5	6	3	-4	-1	5
3	-5	2	1	-6	4	5	-2	-1	6	-4
4	-1	6	-2	5	-3	1	-6	2	-5	3
5	3	-1	6	-4	-2	-3	1	-6	4	2
6	2	-4	-5	3	-1	-2	4	5	-3	1

Figura 5.21 - Exemplo de escala de jogos obtida pelo processo de assimilação.

5.6.3 Componente AA

O componente AA (Analisador de Agrupamentos) é executado toda vez que um cluster é atualizado com algum indivíduo. Quando isso ocorre, o AA verifica se o cluster já pode ser considerado promissor. Além desta tarefa, o analisador de agrupamentos realiza um *esfriamento* de todos os *clusters* que foram ativados a cada geração e elimina aqueles que não foram ativados.

O processo de *esfriamento* consiste em fazer com que o *cluster* retorne ao seu estado inicial, como se ainda não houvesse ocorrido nenhuma ativação. *Clusters esfriados* podem, eventualmente, ser *reaquecidos* através da seleção de indivíduos de sua região de busca. Além disso, todos os *clusters* que permanecem inativos ou com baixa densidade durante uma geração são eliminados. A eliminação dos *clusters* com baixa densidade permite que outros centros sejam descobertos. Segundo Oliveira (2004), pode-se fazer uma analogia entre esse processo e os ecossistemas naturais, onde os indivíduos em cada região interagem com o meio-ambiente, promovendo uma forma de exploração desse meio. Uma vez esgotados os recursos naturais de uma região, os indivíduos podem migrar para outras regiões ainda não exploradas.

Um *cluster* se torna promissor quando atinge uma certa densidade λ_t, dada por:

$$\lambda_t \geq PD \left[\frac{NS}{NT_c} \right] \tag{5.4}$$

onde PD indica quantas vezes a densidade deve estar acima do normal para que um *cluster* seja considerado promissor, NS o número de soluções (indivíduos) que, a cada geração, são selecionados e NT_c o número total de *clusters*. Neste trabalho, foram utilizados os valores $NS = 200$, $NT_c = 20$ e $PD = 2$.

5.6.4 Componente AO

O componente AO (algoritmo de otimização) foi implementado tomando como base a técnica *Iterated Local Search* (ILS), descrita na Subseção 4.3.4. É um método que procura focar sua busca num subespaço definido por soluções que são ótimas locais de um determinado mecanismo de otimização. Por esta característica, a aplicação da técnica em um indivíduo que representa o centro de uma possível região promissora, se apresenta como uma solução interessante e viável.

Tomando então uma solução inicial localizada em um ótimo local, o algoritmo realiza uma perturbação nesta solução, seguida pela aplicação de uma busca local. A perturbação adotada neste trabalho foi a aplicação do movimento Troca Jogos, pela sua importância na obtenção de soluções não alcançadas por outros movimentos.

A técnica utilizada na etapa de busca local é similar a metaheurística *VNS* (Subseção 4.3.3). Neste algoritmo são utilizadas três estruturas de vizinhança, sendo elas definidas pelos movimentos TT, TPR e TMC. Inicialmente, explora-se a vizinhança TT. Uma vez que um ótimo local para esta vizinhança seja encontrado, é aplicado sobre a escala uma busca rápida pela melhor atribuição de mandos de campo, utilizando o movimento TMC. Em seguida, o movimento TPR é investigado e, como no caso anterior, uma vez localizado um ótimo local, este é submetido à busca pela melhor atribuição de mandos de campo. Estas operações se repetem até que um ótimo local referente às três estruturas de vizinhanças seja encontrado. A geração de soluções vizinhas foi realizada de duas formas. Na primeira, elas são visitadas de forma aleatória dentro de cada vizinhança. Esse processo é bem rápido, além de exigir pouco esforço computacional. Na segunda, é realizada uma busca mais exaustiva dentro das vizinhanças, percorrendo toda a escala e fazendo as alterações referentes ao movimento corrente em cada iteração. Esse processo se mostrou um pouco mais eficaz que o primeiro, porém muito demorado e com custo computacional alto, uma vez que é necessário o cálculo da função de avaliação para cada vizinho gerado.

A solução obtida pela busca local pode ou não ser aceita como nova solução corrente, dependendo do atendimento do critério de aceitação. Este critério segue a seguinte lógica. Foi definido um parâmetro α, que é utilizado para relaxar o critério de aceitação, possibilitando ao algoritmo uma maior exploração do espaço de busca. Se uma solução s' for menor que $(1 + \alpha)$ vezes o valor da melhor solução s_c, ela passa a ser a nova solução corrente. O valor de α é iniciado com 0.0001 e a cada iteração sem melhora ele é aumentado em 0.05% sob seu valor corrente. Quando uma solução é aceita como nova solução corrente o valor de α é reiniciado.

Outra alteração realizada na forma tradicional do método ILS diz respeito a reinicialização da solução corrente. Para isso, durante o procedimento, é armazenada a segunda melhor solução encontrada (s'_2). Se após cinqüenta iterações o ILS não obtiver melhora na solução, o processo é reiniciado, tomando s'_2 como nova solução corrente. Essa reinicialização é realizada no máximo três vezes.

A Figura 5.22 apresenta o pseudocódigo do algoritmo *ILS-AO*, onde $IterMax$ é o número máximo de iterações realizadas pelo algoritmo, sendo definido como 1000.

```
Procedimento ILS-AO(s*)
1   s_c ← s*;                          { Solução corrente s_c }
2   IterSM ← 0;                        { Iterações sem melhora }
3   NR ← 0;                            { Número de reinicializações }
4   Iter ← 0;                          { Iteração corrente }
5   α ← 0.0001;                        { Parâmetro de relaxamento }
6   s'_2 ← Solução aleatória;          { Inicializa s'_2 }
7   enquanto (Iter < IterMax & NR ≤ 3) faça
8       s ← Perturbacao(s_c);
9       s' ← BuscaLocal(s);
10      se (f(s') < (f(s_c) * (1 + α))) então
11          s_c ← s';
12          α ← 0.0001;
13          IterSM ← 0;
14      senão
15          α ← α * 0.05%;
16          IterSM ← IterSM + 1;
17          se (f(s') < f(s'_2)) então
18              s'_2 ← s';
19          fim-se;
20      fim-senão;
21      se (IterSM > 50) então
22          s_c ← s'_2;
23          IterSM ← 0;
24          NR ← NR + 1;
25      fim-se;
26      Iter ← Iter + 1;
27  fim-enquanto;
28  Retorne s_c;
fim-ILS-AO;
```

Figura 5.22 - Algoritmo *ILS-AO*

5.6.5 Integração dos Componentes do ECS

A integração entre os quatro componentes do ECS é apresentada na Figura 5.23. A cada iteração i dois indivíduos são selecionados e agrupados pelo componente AI (linha 8), que retorna um *cluster* C_{ativo} para o componente AA (linha 9), que irá analisar se C_{ativo} pode ser considerado promissor. Se o *cluster* for considerado promissor ($CP = Verdadeiro$), ele é explorado pelo componente AO (linha 11).

```
Procedimento ECS(.)
1   Inicialize a população P;
2   C_ativo ← ∅;          { Define se um cluster está ativo }
3   CP ← Falso;           { Define se um cluster é promissor }
4    enquanto (g < nGeracoes) faça
5        enquanto (i < nIndividuos) faça
6            pai1 ← Selecione um individuo ∈ 70% melhores de P;
7            pai2 ← Selecione um individuo ∈ 30% piores de P;
8            C_ativo ← ComponenteAI(pai1, pai2);
9            CP ← ComponenteAA(C_ativo);
10           se(CP = Verdadeiro) então
11               ComponenteAO(C_ativo);
12           filho ← Recombinacao(pai1, pai2);
13           se (atende critério de mutacao) então
14               Mutacao(filho);
15           fim-se
16           filho' ← BuscaLocal(filho);
17           Avaliacao(filho');
18           Adicione filho' em P;
19           i ← i + 1;
20       fim-enquanto
21       P ← DefinaSobreviventes(P);
22       g ← g + 1;
23   fim-enquanto
fim-ECS
```

Figura 5.23 - Algoritmo ECS

Após a execução dos componentes do ECS os indivíduos selecionados são recombinados e o filho gerado pode sofrer mutação. Este filho vai para a população de indivíduos e poderá então, se selecionado, fazer parte do processo de agrupamento de soluções.

5.7 *CS Aplicado ao mTTP

O *Clustering Search* (*CS) é uma adaptação do ECS que vem sendo utilizado com sucesso recentemente (OLIVEIRA; LORENA, 2006b; OLIVEIRA; LORENA, 2006a). No *CS o algoritmo evolutivo do componente AE é substituído por outra metaheurística, sendo que esta precisa ser capaz de gerar um número suficiente de soluções diferentes para o processo de agrupamento. Neste sentido, utilizou-se a metaheurística *Variable Neighborhood Search* (VNS) como alternativa, gerando a abordagem *VNS-CS*. A Figura 5.24 apresenta o diagrama conceitual da abordagem *CS. Repare que o componente AE foi substituído por um outro componente, chamado de ME.

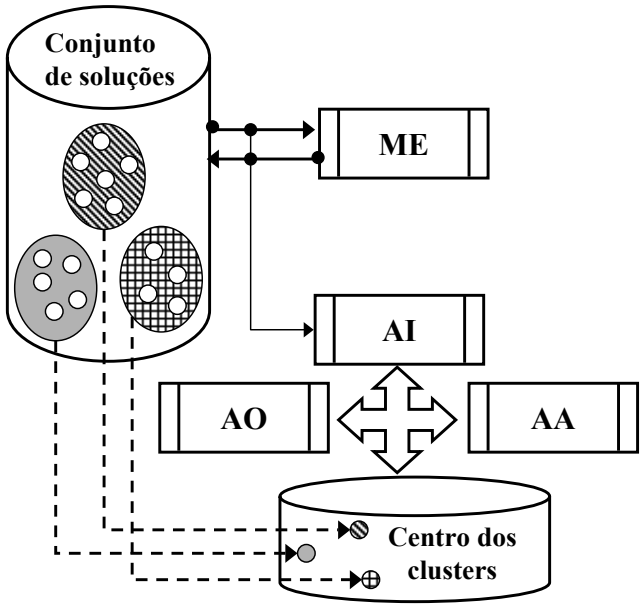

Figura 5.24 - Diagrama Conceitual do *CS.

Fonte: Oliveira e Lorena (2006a)

Conforme já descrito na Subseção 4.3.3, o VNS é um método de busca local que realiza uma exploração cada vez mais distante da solução corrente, pois sua busca consiste em explorar o espaço de soluções através de trocas sistemáticas nas estruturas de vizinhança. Foram utilizadas três estruturas de vizinhanças diferentes na implementação do VNS, na seguinte ordem: TJ, TR e TPR. Cada nova solução s' gerada dentro de uma das $N^{(k)}(s)$ estruturas de vizinhança é submetida a um método de busca local. Para esta etapa utilizou-se o método *Variable Neighborhood Descent* (VND) com as outras duas estruturas de vizinhança, dispostas na seguinte ordem: TT e TMC.

Cada solução gerada pelo componente ME é atribuída ao componente AI (Subseção 5.6.2),

responsável por fazer a classificação da solução e então atribuí-la ao *cluster* mais similar. A implementação utilizada no *CS é a mesma que foi utilizada no ECS, uma vez que elas possuem a mesma função.

Assim como o componente AI, os demais componentes do *CS (AA e AO) foram implementados da mesma forma. A única diferença está no componente AA, onde o valor de NS (número de soluções selecionadas) na Equação 5.4 foi definido, por simplicidade, com o valor 300, ou seja, 10% do número máximo de soluções que o algoritmo pode gerar para o processo de agrupamento.

A Figura 5.25 apresenta o pseudocódigo do algoritmo VNS-CS. Este procedimento é executado enquanto o valor de $IterMax$ não for atingido (linha 5). Neste trabalho utilizou-se $IterMax = 1000$. A cada iteração do VNS uma solução é gerada, sendo em seguida agrupada pelo ComponenteAI (linha 10) em algum *cluster* C_{ativo}, que será analisado pelo ComponenteAA (linha 11). Se o *cluster* for considerado promissor ($CP = Verdadeiro$), ele é explorado pelo ComponenteAO (linha 13).

```
Procedimento VNS-CS(f(.), N(.), s_0)
1  Iter ← 0;            { Iteração corrente }
2  C_ativo ← ∅;         { Define se um cluster está ativo }
3  CP ← Falso;          { Define se um cluster é promissor }
4  s ← s_0;             { Solução corrente }
5  enquanto (Iter < IterMax) faça
6      k ← 1;
7      enquanto (k ≤ 3) faça
8          Gere um vizinho qualquer s' ∈ N_(k)(s);
9          s'' ← VND(s');
10         C_ativo ← ComponenteAI(s'');
11         CP ← ComponenteAA(C_ativo);
12         se(CP = Verdadeiro) então
13             ComponenteAO(C_ativo);
14         se (f(s'') < f(s)) então
15             s ← s'';
16             k ← 1;
17         senão
18             k ← k + 1;
19         fim-se;
20     fim-enquanto;
21     Iter ← Iter + 1;
22 fim-enquanto;
fim-VNS-CS;
```

Figura 5.25 - Algoritmo *VNS-CS*

6 EXPERIMENTOS COMPUTACIONAIS e RESULTADOS

Os algoritmos apresentados nas seções anteriores (*GA-SA*, *ECS* e *VNS-CS*) foram codificados em linguagem C++ e compilados com a versão 5.0 do *Borland C++ Builder*. Todos os testes foram conduzidos em um computador *Pentium IV 3.0 GHz com 512 MBytes de memória RAM*.

6.1 Instâncias de Teste Utilizadas

Neste trabalho foram utilizadas três classes de instâncias teste, disponíveis em TOURN *website*.

A primeira classe é composta por instâncias circulares. Tais instâncias foram geradas para testar se os problemas relacionados à geração de escalas de jogos são mais fáceis de resolver que os problemas do caixeiro viajante, uma vez que para este último as instâncias circulares são de simples resolução. A notação utilizada para cada instância é CIRCn, onde $n = 4, 6, 8, 10, 12, 14, 16, 18,$ e 20 times. Os nós são dispostos numa "circunferência" de forma que a distância de cada nó aos seus vizinhos seja igual a 1. Eles são numerados crescentemente de $0, 1, 2,$ ...,$(n-1)$, sendo a distância entre dois nós, i e j, onde $i > j$, dada pelo comprimento do caminho mais curto entre eles, ou seja, igual ao mínimo entre $(i-j)$ e $(j-i+n)$. A Figura 6.1 apresenta a formatação do arquivo contendo a instância CIRC4.

0	1	2	1
1	0	1	2
2	1	0	1
1	2	1	0

Figura 6.1 - Instância CIRC4

A segunda classe de instâncias foi criada a partir de um subconjunto de times da MLB dos Estados Unidos (*National League*). Foram consideradas as distâncias entre as cidades desses times, gerando as instâncias NLn, onde $n = 4, 6, 8, 10, 12, 14$ e 16 times. A Figura 6.2 apresenta a formatação do arquivo contendo a instância NL4.

0	745	665	929
745	0	80	337
665	80	0	380
929	337	380	0

Figura 6.2 - Instância NL4

A terceira classe utilizada é conhecida como classe de *instâncias constantes* e foi definida em Ribeiro e Urrutia (2006a). Tal classe de instâncias foi criada com o objetivo de provar a otimalidade de soluções encontradas para as instâncias de minimização de distâncias. Para cada uma das instâncias CONn, onde $n = 4, 6, 8, 10, 12, 14, 16, 18$, e 20 times, a distância entre um nó e cada um dos outros é igual a 1. A Figura 6.3 apresenta a formatação do arquivo contendo a instância CON4.

$$\begin{matrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{matrix}$$

Figura 6.3 - Instância CON4

Além dessas classes descritas, será utilizada uma instância originada a partir dos times que participaram do Campeonato Brasileiro de Futebol no ano 2003. Esta edição do Campeonato Brasileiro contou com a participação de 24 times, portanto, será a maior instância a ser tratada neste trabalho, sendo conhecida por *br2003.24* e também disponível em TOURN *website*.

6.2 Resultados Computacionais

As próximas seções apresentam os resultados obtidos por cada um dos algoritmos propostos neste trabalho. As tabelas que apresentam os resultados seguem a seguinte formatação: na primeira coluna são apresentadas as instâncias testadas, na segunda os melhores resultados da literatura, obtidos pelos trabalhos de Ribeiro e Urrutia (2004) e Hentenryck e Vergados (2006), na terceira os resultados obtidos pelos algoritmos propostos e nas duas últimas o *gap* e o tempo de processamento necessário para se obter cada uma das soluções, respectivamente. Os parâmetros utilizados pelos algoritmos foram definidos de forma empírica, após uma bateria preliminar de testes. Tais parâmetros são apresentados no Apêndice A.

6.2.1 Resultados Computacionais do Algoritmo GA-SA

As tabelas que se seguem apresentam os melhores resultados obtidos pelo algoritmo *GA-SA* em cinco execuções do algoritmo para cada uma das instâncias apresentadas na Seção 6.1.

Tabela 6.1 - Resultados obtidos pelo *GA-SA* para as instâncias CIRCn

Instâncias	Literatura	Obtidos	gap (%)	Tempo (seg)
CIRC4	20	20	0%	2
CIRC6	72	72	0%	4
CIRC8	140	142	1,41%	48
CIRC10	272	282	3,55%	355
CIRC12	432	458	5,68%	1142
CIRC14	696	714	2,52%	26
CIRC16	968	980	1,22%	437
CIRC18	1306	1370	4,67%	2029
CIRC20	1852	1882	1,59%	568

Tabela 6.2 - Resultados obtidos pelo *GA-SA* para as instâncias NLn

Instâncias	Literatura	Obtidos	gap (%)	Tempo (seg)
NL4	8276	8276	0%	2
NL6	26588	26588	0%	3
NL8	41928	43025	2,55%	867
NL10	63832	66264	3,67%	130
NL12	119608	120981	1,13%	1066
NL14	199363	208086	4,19%	356
NL16	279077	290188	3,83%	865

Os resultados apresentados na Tabela 6.1, Tabela 6.2, Tabela 6.3 e Tabela 6.4 demonstram que a metodologia utilizada é competitiva, uma vez que os resultados estão bem próximos aos melhores conhecidos, chegando em algumas instâncias, a zerar *gap*. Em todas as instâncias testadas os piores resultados se distanciam dos melhores da literatura em apenas 5,68% para as instâncias CIRCn, 4,19% para as instâncias NLn e 1,56% para as instâncias CONn.

O resultado obtido para a instância que representa o problema real do Campeonato Brasileiro de Futebol se distancia em apenas 1,58% do melhor resultado conhecido na literatura. Se compararmos este resultado com o apresentado pela escala real utilizada na edição de 2003 deste campeonato, a economia chegaria em 51,2% da distância de viagem praticada naquela edição. Analisando este resultado em termos de distância, na escala oficial os times percorreram 1.048.134 km e na escala obtida por este trabalho esta distâncias foi reduzida para 511.256 km.

O tempo de processamento para se obter cada um dos resultados apresentados variou de 2 a 2029 segundos, o que representa, portanto, um processamento computacional viável.

Tabela 6.3 - Resultados obtidos pelo *GA-SA* para as instâncias CON*n*

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
CON4	17	17	0%	2
CON6	48	48	0%	3
CON8	80	81	1,23%	28
CON10	130	130	0%	75
CON12	192	193	0,52%	343
CON14	252	256	1,56%	201
CON16	342	343	0,29%	596
CON18	432	436	0,92%	44
CON20	520	526	1,14%	104

Tabela 6.4 - Resultados obtidos pelo *GA-SA* para a instância br2003.24

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
br2003.24	503158	511256	1,58%	1706

Com o intuito de medir o desempenho do algoritmo, foram realizadas outras 4 execuções para cada uma das instâncias testadas. A partir dos resultados obtidos, um cálculo de desvio foi realizado. Para calcular este desvio foi utilizada a seguinte equação:

$$Desvio = \left(\frac{DistMedia - DistMenor}{DistMenor} \right) \times 100 \tag{6.1}$$

onde *DistMedia* representa a média aritmética das distâncias obtidas nos 5 experimentos e *DistMenor* a menor distância obtida. Os valores de desvio obtidos se encontram dentro do intervalo $0\% < Desvio < 1,05\%$, sendo o maior desvio obtido para a instância br2003.24. A Tabela 6.5 apresenta os resultados obtidos e utilizados para o cálculo deste desvio.

Tabela 6.5 - Cálculo do desvio para a instância br2003.24

Instância br2003.24				
Exe1	Exe2	Exe3	Exe4	Exe5
514654	527225	514304	**511256**	515776
Desvio = 1,05%				

O algoritmo *GA-SA* apresentou para todas as instâncias um desvio muito pequeno ($0\% < Desvio < 1,05\%$), demonstrando o bom desempenho da abordagem proposta.

6.2.2 Resultados Computacionais do Algoritmo ECS

As tabelas que se seguem apresentam os melhores resultados obtidos pelo algoritmo ECS em cinco execuções do algoritmo para cada uma das instâncias apresentadas na Seção 6.1.

Tabela 6.6 - Resultados obtidos pelo ECS para as instâncias CIRC*n*

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
CIRC4	20	20	0%	2
CIRC6	72	72	0%	3
CIRC8	140	140	0%	125
CIRC10	272	278	2,16%	2397
CIRC12	432	458	5,68%	578
CIRC14	696	714	2,52%	148
CIRC16	968	980	1,22%	208
CIRC18	1306	1380	5,36%	790
CIRC20	1852	1882	1,59%	182

Tabela 6.7 - Resultados obtidos pelo ECS para as instâncias NL*n*

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
NL4	8276	8276	0%	2
NL6	26588	26588	0%	5
NL8	41928	42034	0,25%	2831
NL10	63832	65314	2,27%	4285
NL12	119608	120981	1,13%	1037
NL14	199363	208086	4,19%	254
NL16	279077	290188	3,83%	1471

Os resultados apresentados na Tabela 6.6, Tabela 6.7, Tabela 6.8 e Tabela 6.9 demonstram que o ECS se apresenta como solução heurística competitiva. Primeiramente, pelo fato de, utilizando o algoritmo *GA-SA* como componente AE, foi possível melhorar boa parte dos resultados da abordagem que utiliza o *GA-SA* de forma isolada. Outro fato é que alguns resultados estão bem mais próximos aos melhores conhecidos na literatura. Em algumas instâncias o *gap* foi zerado. Em todas as instâncias testadas os piores resultados se distanciam dos melhores da literatura em apenas 5,68% para as instâncias CIRC*n*, 4,19% para as instâncias NL*n* e 1,56% para as instâncias CON*n*, ou seja, resultado semelhante ao apresentado pela abordagem *GA-SA*. Dessa forma, fica clara a importância do componente AE do ECS, pois ele pode influenciar diretamente no desempenho final do método.

Tabela 6.8 - Resultados obtidos pelo ECS para as instâncias CON*n*

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
CON4	17	17	0%	2
CON6	48	48	0%	4
CON8	80	81	1,23%	17
CON10	130	130	0%	52
CON12	192	193	0,52%	406
CON14	252	256	1,56%	198
CON16	342	343	0,29%	338
CON18	432	434	0,46%	739
CON20	520	526	1,14%	291

Tabela 6.9 - Resultados obtidos pelo ECS para a instância br2003.24

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
br2003.24	503158	511256	1,58%	938

Para a instância do Campeonato Brasileiro de Futebol o resultado obtido foi idêntico ao obtido pelo *GA-SA*, porém em tempo computacional 45% menor. Esse fato indica, claramente, que o ECS diminuiu consideravelmente o esforço computacional exigido pela aplicação do *GA-SA*. Esse fato reforça a idéia de que, a localização de possíveis regiões promissoras pode agilizar o processo de busca por boas soluções.

O tempo de processamento para se obter cada um dos resultados apresentados representa um processamento computacional viável, variando de 2 a 4285 segundos, sendo o maior, o tempo para se encontrar a melhor solução para a instância NL10.

Assim como na análise do *GA-SA*, foram realizadas outras 4 execuções para cada uma das instâncias testadas, com o intuito de avaliar o desempenho do algoritmo ECS. A partir dos resultados obtidos, um cálculo de desvio foi realizado utilizando a Equação 6.1. Os valores de desvio obtidos se encontram dentro do intervalo $0\% < Desvio < 1,58\%$, sendo o maior desvio calculado o da instância NL10. A Tabela 6.10 apresenta os resultados obtidos e utilizados para o cálculo do desvio para a instância NL10.

Tabela 6.10 - Cálculo do desvio para a instância NL10

Instância NL10				
Exe1	Exe2	Exe3	Exe4	Exe5
66997	**65314**	66900	66264	66264
Desvio = 1,58%				

Assim como na abordagem anterior, o algoritmo ECS apresentou para todas as instâncias um desvio muito pequeno ($0\% < Desvio < 1,58\%$), demonstrando o bom desempenho desta abordagem.

6.2.3 Resultados Computacionais do Algoritmo VNS-CS

Nesta seção são apresentados os melhores resultados obtidos pelo algoritmo *VNS-CS* em cinco execuções do algoritmo para cada uma das instâncias apresentadas na Seção 6.1.

Tabela 6.11 - Resultados obtidos pelo *VNS-CS* para as instâncias CIRC*n*

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
CIRC4	20	20	0%	6
CIRC6	72	72	0%	9
CIRC8	140	140	0%	438
CIRC10	272	276	1,45%	4951
CIRC12	432	446	3,14%	1275
CIRC14	696	702	0,85%	3672
CIRC16	968	978	1,02%	826
CIRC18	1306	1352	3,40%	1153
CIRC20	1852	1882	1,59%	1189

Tabela 6.12 - Resultados obtidos pelo *VNS-CS* para as instâncias NL*n*

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
NL4	8276	8276	0%	5
NL6	26588	26588	0%	7
NL8	41928	41928	0%	1030
NL10	63832	65193	2,09%	228
NL12	119608	120906	1,07%	2630
NL14	199363	208824	4,53%	769
NL16	279077	287130	2,80%	3201

Os resultados obtidos pela abordagem *VNS-CS* e apresentados na Tabela 6.11, Tabela 6.12, Tabela 6.13 e Tabela 6.14 demonstram que a utilização conjunta das técnicas VNS e *Clustering Search* (CS) pode resultar em soluções de boa qualidade. Especificamente neste trabalho, a utilização conjunta destes métodos conseguiu superar as outras duas abordagens propostas em várias instâncias, tornando os resultados bem mais próximos aos melhores conhecidos na literatura. Em algumas instâncias os resultados obtidos foram iguais aos da literatura, ou seja, o *gap* foi zerado. Em todas as instâncias

Tabela 6.13 - Resultados obtidos pelo *VNS-CS* para as instâncias CONn

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
CON4	17	17	0%	4
CON6	48	48	0%	6
CON8	80	81	1,23%	39
CON10	130	130	0%	92
CON12	192	193	0,52%	299
CON14	252	255	1,18%	131
CON16	342	343	0,29%	407
CON18	432	433	0,23%	1265
CON20	520	525	0,95%	1102

Tabela 6.14 - Resultados obtidos pelo *VNS-CS* para a instância br2003.24

Instâncias	Literatura	Obtidos	*gap* (%)	Tempo (*seg*)
br2003.24	503158	512545	1,83%	798

testadas os piores resultados se distanciam dos melhores da literatura em apenas 3,40% para as instâncias CIRCn, 4,53% para as instâncias NLn e 1,23% para as instâncias CONn. Nota-se que, o desempenho final do *VNS-CS* foi melhor que o apresentado pelos métodos *GA-SA* e ECS.

Para a instância do Campeonato Brasileiro de Futebol o resultado obtido foi um pouco inferior que o obtido pelas outras abordagens, com *gap* de 0,25% em relação a estes resultados. Em contrapartida, o tempo de processamento para se obter tal resultado foi menor que o tempo apresentado pelas demais abordagens. Além disso, a economia em relação à escala oficial da edição 2003 do Campeonato Brasileiro de Futebol continuou com valor considerável, cerca de 51,09% menor, em termos de deslocamento dos times.

O tempo de processamento para se obter cada um dos resultados apresentados, apesar de serem um pouco superiores que das abordagens anteriores, representa um processamento computacional viável, variando de 4 a 4951 segundos. Este último, foi o tempo necessário para se encontrar a melhor solução para a instância CIRC10.

Assim como nas análises anteriores, foram realizadas outras 4 execuções para cada uma das instâncias testadas, para avaliar o desempenho do algoritmo *VNS-CS*. O cálculo de desvio foi realizado utilizando a Equação 6.1. Os valores de desvio obtidos se encontram dentro do intervalo $0\% < Desvio < 1,88\%$, sendo o maior desvio calculado o da instância CIRC10. A Tabela 6.15 apresenta os resultados obtidos e utilizados para o cálculo do

desvio para a instância CIRC10.

Tabela 6.15 - Cálculo do desvio para a instância CIRC10

Instância CIRC10				
Exe1	Exe2	Exe3	Exe4	Exe5
278	282	282	288	**276**
Desvio = 1,88%				

Repetindo o desempenho das abordagens anteriores, o algoritmo *VNS-CS* apresentou para todas as instâncias um desvio muito pequeno ($0\% < Desvio < 1,88\%$), ou seja, demonstrando a robustez do algoritmo quando utilizado na resolução do mTTP.

6.3 Comparação das Abordagens

Conforme apresentado nas seções anteriores, os resultados obtidos em cada uma das três abordagens propostas foram satisfatórios, chegando bem próximo aos melhores resultados existentes na literatura. Nesta seção será realizado um comparativo entre as abordagens, na tentativa de identificar qual foi a mais eficiente na resolução do mTTP.

Apesar de serem abordagens distintas, todas as três possuem algum ponto de ligação entre si. Por exemplo, a técnica ECS faz uso do *GA-SA* no componente AE. Já a técnica *VNS-CS* apenas substitui o componente AE do ECS pela metaheurística VNS, mantendo os demais componentes quase intactos (ver Seção 5.7). Portanto, o objetivo do comparativo entre as abordagens é ajudar a identificar quais os pontos positivos e negativos de cada técnica.

As tabelas a seguir apresentam os resultados obtidos por cada uma das técnicas, sendo apresentado em negrito o melhor resultado para cada uma das instâncias testadas. A primeira coluna apresenta as instâncias testadas, na segunda os resultados do *GA-SA*, na terceira os resultados do ECS, na quarta os resultados do *VNS-CS* e na última o tempo de processamento necessário para se obter a melhor solução entre as três abordagens. Quando houver empate entre duas ou mais abordagens, será considerado melhor resultado o valor obtido em menor tempo computacional.

Analisando o comparativo apresentado na Tabela 6.16, Tabela 6.17, Tabela 6.18 e Tabela 6.19, verifica-se que as abordagens ECS e *VNS-CS* foram, em quase todas as instâncias testadas, melhores que a abordagem *GA-SA*. Fazendo um balanço final, temos que o método *GA-SA* foi melhor em 2 instâncias, o ECS foi melhor em 11 e o *VNS-CS* foi melhor em 13. Outro ponto importante é que, em apenas duas instâncias (NL14 e

br2003.24) o *VNS-CS* foi superado em termos de resultado, ou seja, das 26 instâncias testadas o *VNS-CS* foi melhor ou igual aos resultados obtidos pelos outros métodos em 24 instâncias.

Tabela 6.16 - Resultados obtidos pelo *VNS-CS* para as instâncias CIRC*n*

Instâncias	*GA-SA*	ECS	*VNS-CS*	Tempo (*seg*)
CIRC4	20	**20**	20	2
CIRC6	72	**72**	72	3
CIRC8	142	**140**	140	125
CIRC10	282	278	**276**	4951
CIRC12	458	458	**446**	1275
CIRC14	714	714	**702**	3672
CIRC16	980	980	**978**	826
CIRC18	1370	1380	**1352**	1153
CIRC20	1882	**1882**	1882	182

Tabela 6.17 - Resultados obtidos pelo *VNS-CS* para as instâncias NL*n*

Instâncias	*GA-SA*	ECS	*VNS-CS*	Tempo (*seg*)
NL4	8276	**8276**	8276	2
NL6	**26588**	26588	26588	3
NL8	43025	42034	**41928**	1030
NL10	66264	65314	**65193**	228
NL12	120981	120981	**120906**	2630
NL14	208086	**208086**	208824	254
NL16	290188	290188	**287130**	3201

Este resultado talvez seja justificado pelo fato de toda solução gerada pelo *GA-SA* que não sofre mutação, se encontra dentro do molde imposto pelo Método do Polígono. Por este fato, altos índices de mutação (movimento Troca Jogos) são necessários para que a solução seja "retirada" deste padrão de soluções, aumentando o espaço de busca. Dessa forma, o *VNS-CS* leva vantagem em relação às outras duas abordagens, pois toda solução gerada é submetida ao movimento Troca Jogos, possibilitando então uma busca num espaço maior de soluções.

Realizando um comparativo entre os melhores resultados obtidos neste trabalho e os melhores resultados da literatura (para o mTTP) obtidos pelas técnicas *Grils-mTTP* e *TTSA*, apresentadas pelos trabalhos de Ribeiro e Urrutia (2004) e Hentenryck e Vergados

Tabela 6.18 - Resultados obtidos pelo *VNS-CS* para as instâncias CON*n*

Instâncias	*GA-SA*	ECS	*VNS-CS*	Tempo (*seg*)
CON4	17	**17**	17	2
CON6	**48**	48	48	3
CON8	81	**81**	81	17
CON10	130	**130**	130	52
CON12	193	193	**193**	299
CON14	256	256	**255**	131
CON16	343	**343**	343	338
CON18	436	434	**433**	1265
CON20	526	526	**525**	1102

Tabela 6.19 - Resultados obtidos pelo *VNS-CS* para a instância br2003.24

Instâncias	*GA-SA*	ECS	*VNS-CS*	Tempo (*seg*)
br2003.24	511256	**511256**	512545	938

(2006), respectivamente, é possível se realizar algumas análises interessantes. Nas próximas tabelas serão realizados estes comparativos.

As células com valor "-" indicam que os respectivos valores não foram divulgados ou abordados pelo autor. Neste comparativo, apenas as instâncias CIRC e NL serão utilizadas, pois elas foram abordadas pelos dois trabalhos alvo desse comparativo.

Tabela 6.20 - Comparação com Ribeiro e Urrutia (2004) - Instâncias CIRC*n*. Entre parênteses, apresenta-se a abordagem que obteve o melhor resultado deste trabalho. 1: *GA-SA*; 2: ECS; 3: *VNS-CS*

Instâncias	Literatura	Tempo Lit.	Melhores	Tempo (*seg*)
CIRC4	20	-	20 (2)	2
CIRC6	72	-	72 (2)	3
CIRC8	140	-	140 (2)	125
CIRC10	272	-	276 (3)	4951
CIRC12	456	-	**446** (3)	1275
CIRC14	714	-	**702** (3)	3672
CIRC16	978	-	978 (3)	826
CIRC18	1306	-	1352 (3)	1153
CIRC20	1882	-	1882 (2)	182

Conforme apresentado na Tabela 6.20 e na Tabela 6.21 os melhores resultados obtidos pelas abordagens propostas neste trabalho estão no mesmo nível dos melhores resultados

Tabela 6.21 - Comparação com Ribeiro e Urrutia (2004) - Instâncias NL*n*. Entre parênteses, apresenta-se a abordagem que obteve o melhor resultado deste trabalho. 1: *GA-SA*; 2: ECS; 3: *VNS-CS*

Instâncias	Literatura	Tempo Lit.	Melhores	Tempo (seg)
NL4	8276	-	8276 (2)	2
NL6	26588	-	26588 (1)	3
NL8	41928	-	41928 (3)	1030
NL10	63832	-	65193 (3)	228
NL12	120665	-	120906 (3)	2630
NL14	208086	-	208086 (2)	254
NL16	279618	-	287130 (3)	3201

conhecidos para a heurística *Grils-mTTP*, apresentados no trabalho de Ribeiro e Urrutia (2004). Além disso, para as instâncias CIRC12 e CIRC14 os resultados obtidos pela abordagem *VNS-CS* foram superiores que os obtidos pela *Grils-mTTP*. Das 16 instâncias comparadas, os resultados deste trabalho são inferiores em apenas 5 e iguais em outras 9 instâncias. Em relação ao tempo computacional os trabalhos não apresentam quase nenhuma diferença, sendo no trabalho da literatura o tempo máximo de processamento fixado em 15 minutos, enquanto nesta proposta a média entre os tempos necessários para se encontrar as melhores soluções apresentadas foi de $15,18$ minutos.

Outro trabalho que apresenta alguns dos melhores resultados para o mTTP conhecidos na literatura é o de Hentenryck e Vergados (2006). A Tabela 6.22 e Tabela 6.23 apresentam um comparativo entre os melhores resultados das abordagens propostas e os encontrados na literatura.

Tabela 6.22 - Comparação com Hentenryck e Vergados (2006) - Instâncias CIRC*n*. Entre parênteses, apresenta-se a abordagem que obteve o melhor resultado deste trabalho. 1: *GA-SA*; 2: ECS; 3: *VNS-CS*

Instâncias	Literatura	Tempo Lit.	Melhores	Tempo (seg)
CIRC4	-	-	20 (2)	2
CIRC6	-	-	72 (2)	3
CIRC8	140	0.2	140 (2)	125
CIRC10	272	28160	276 (3)	4951
CIRC12	432	93.1	446 (3)	1275
CIRC14	696	53053.5	702 (3)	3672
CIRC16	968	38982.7	978 (3)	826
CIRC18	1352	178997.5	1352 (3)	1153
CIRC20	1852	59097.9	1882 (2)	182

Tabela 6.23 - Comparação com Hentenryck e Vergados (2006) - Instâncias NL*n*. Entre parênteses, apresenta-se a abordagem que obteve o melhor resultado deste trabalho. 1: *GA-SA*; 2: ECS; 3: *VNS-CS*

Instâncias	Literatura	Tempo Lit.	Melhores	Tempo (seg)
NL4	-	-	8276 (2)	2
NL6	-	-	26588 (1)	3
NL8	41928	0.1	41928 (3)	1030
NL10	63832	477.2	65193 (3)	228
NL12	119608	15428.1	120906 (3)	2630
NL14	199363	34152.3	208086 (2)	254
NL16	279077	55640.8	287130 (3)	3201

O ponto chave na comparação deste trabalho com o de Hentenryck e Vergados (2006) é o tempo de processamento computacional. Observa-se que, em tempo computacional muito inferior, as abordagens propostas neste trabalho chegaram a resultados bem próximos aos obtidos pelo TTSA, que exige um enorme esforço computacional, visto que algumas soluções foram encontradas após dias de processamento em um computador *AMD Athlon 64 de 2.0 GHz*. Por exemplo, para a instância CIRC18 o TTSA levou 178997.5 segundos para encontrar sua melhor solução, enquanto o *VNS-CS* levou apenas 1153 segundos para chegar à mesma solução. Além disso, cada instância foi explorada pelo TTSA em 20 execuções do algoritmo, enquanto neste trabalho foram realizadas apenas 5 execuções para cada instância.

Portanto, baseando-se nos resultados apresentados, pode-se concluir que a tarefa de se detectar possíveis regiões promissoras pode diminuir consideravelmente o tempo computacional, uma vez que o espaço de busca pode ser explorado de forma mais racional, conseqüentemente resultando num melhor desempenho.

7 CONCLUSÃO

O problema de Geração de Escalas de Jogos para Torneios Esportivos, ou *Traveling Tournament Problem*, é um problema de grande interesse prático, pois é uma das tarefas mais importantes relacionadas ao planejamento das competições esportivas. Todo rendimento financeiro da competição, bem como os resultados dos jogos, podem ser influenciados diretamente por escalas mal planejadas. Neste trabalho foi realizada uma revisão de alguns trabalhos, voltados ao tratamento do TTP, mTTP e PJCB, existentes na literatura. Além dessa revisão, essa dissertação teve por objetivo a apresentação de três novas abordagens aplicadas na resolução do mTTP (*Mirrored Traveling Tournament Problem*), sendo que estas aplicam metodologias ainda não exploradas na literatura.

A primeira abordagem consiste na utilização conjunta de técnicas evolutivas e busca local. Foi implementado um algoritmo híbrido, associando os Algoritmos Genéticos com a metaheurística *Simulated Annealing*, gerando a abordagem *GA-SA*. O papel do SA nesta implementação foi o direcionamento de cada novo indivíduo gerado pelos processos de seleção e cruzamento do AG, a ótimos locais. Um diferencial da implementação utilizada neste trabalho foi a proposta de uma codificação genética compacta (cromossomos) associada a um algoritmo de expansão de código, responsável por transformar cromossomos em escalas de jogos. Trabalhando dessa forma, o processo evolutivo atuou somente sobre a codificação compacta, junto a qual não há inviabilidade e, conseqüentemente, não há redução da eficácia do método. Além disso, a existência de uma população de candidatos à solução sendo evoluída, juntamente com a aplicação do SA, possibilitou a obtenção de ótimos locais de boa qualidade. Por exemplo, a Figura 7.1 apresenta a linha de evolução da população P mantida pelo *GA-SA* para a instância NL8, no decorrer de 10 gerações.

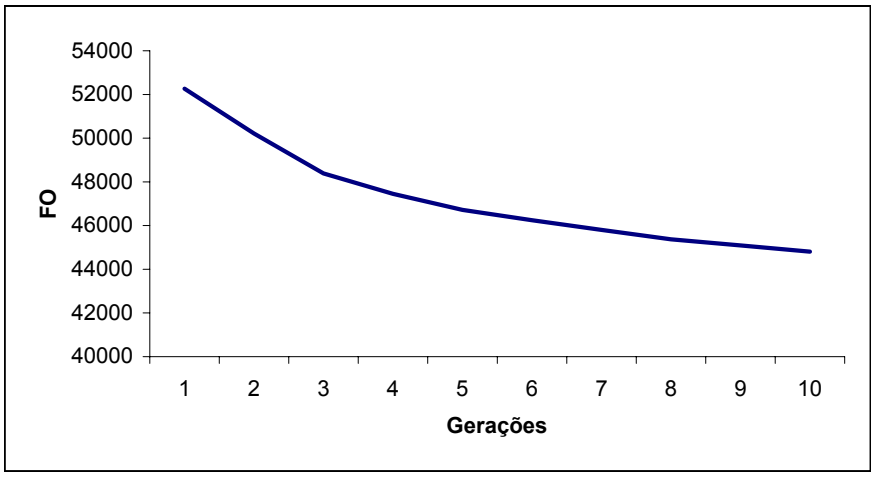

Figura 7.1 - Evolução da população P, mantida pelo *GA-SA* para a instância NL8

Nota-se claramente a evolução da população mantida pelo AG com o passar das gerações. Ao fim do processo de evolução, os resultados obtidos pelo *GA-SA* demonstraram que a técnica híbrida foi robusta e competitiva na resolução do mTTP, quando comparada com outras metaheurísticas apresentadas na literatura. É importante ressaltar que o uso de Algoritmos Genéticos na resolução de problemas de escalonamento de jogos não é muito comum, devido a dificuldade no emprego dos operadores genéticos tradicionais na estrutura das escalas.

A segunda abordagem consiste na aplicação de um algoritmo que realiza uma busca através de agrupamentos. Essa técnica é conhecida na literatura como *Evolutionary Clustering Search* (ECS). Ela faz uso de um algoritmo evolutivo, que funciona como um gerador de soluções, em tempo integral, para o processo de agrupamento. Nesta abordagem, a técnica *GA-SA* foi utilizada para este papel, visando permitir uma comparação entre a primeira e segunda abordagens. A idéia do ECS é a utilização de técnicas de agrupamento para descobrir áreas de busca mais promissoras. Com isso, a estratégia de busca fica mais agressiva quando tais áreas são encontradas, aplicando nestas uma busca local, no caso a técnica utilizada foi o ILS. O objetivo do ECS, portanto, é uma convergência mais rápida por parte do algoritmo, acarretando em possível diminuição do esforço computacional, uma vez que a busca local é realizada de forma mais racional. Essa afirmação pode ser comprovada pela análise dos resultados. Em quase todas as instâncias testadas, o algoritmo conseguiu melhorar o resultado em tempo computacional bem próximo ao gasto pela primeira abordagem. Em contrapartida, o algoritmo apresenta algumas dificuldades. A primeira delas é a grande quantidade de parâmetros a serem calibrados, aumentando sua complexidade, pois conseguir uma sincronia entre esses parâmetros é de suma importância para o seu sucesso. Outra dificuldade apresentada foi a definição da métrica de distância entre soluções. Primeiramente, pela estrutura das escalas de jogos, onde uma alteração mínima pode gerar no final uma escala totalmente diferente. Além disso, chegar a um valor que retrate fielmente o grau de similaridade entre soluções é muito difícil. No entanto, presume-se que a técnica foi empregada com sucesso, uma vez que os resultados obtidos foram competitivos aos apresentados na literatura, além de superar os resultados apresentados pelo *GA-SA*.

A terceira proposta apresentada neste trabalho consiste numa adaptação do ECS, conhecida como *Clustering Search* (*CS). Nesta adaptação, o algoritmo evolutivo do ECS é substituído por outra metaheurística que seja capaz de gerar uma grande quantidade de soluções para o processo de agrupamentos. Neste sentido, foi originada a abordagem *VNS-CS*, pois foi utilizada a técnica *Variable Neighborhood Search* para execução desta etapa. Em relação a técnica ECS as alterações foram poucas, consistindo

apenas na substituição do componente AE pelo componente ME. No entanto, os resultados computacionais foram surpreendentes. Apesar de tempos computacionais maiores que os das outras duas abordagens, o *VNS-CS* conseguiu superar quase todos os resultados apresentados por elas, sendo que, em apenas um ele obteve um resultado inferior. Este resultado pode ser justificado devido a algumas características presentes nos algoritmos evolutivos utilizados nas outras duas abordagens. Tanto no *GA-SA* quanto no ECS, as soluções geradas pelo algoritmo de expansão de código se encontram dentro de um molde imposto pelo Método do Polígono, que é utilizado nesta etapa. Conforme apresentado, as únicas formas de se escapar deste padrão de soluções é a aplicação dos movimentos Troca Parcial de Rodadas e Troca Jogos. Este último foi utilizado como operador de mutação dos algoritmos evolutivos, daí a necessidade de se manter altos índices de mutação nestes algoritmos. Portanto, fica evidente a dependência que estes métodos possuem em relação ao operador de mutação. Porém, a aplicação deste operador em todos os filhos gerados tornaria o processo de evolução muito lento, além de não garantir melhora significativa. Já no algoritmo VNS, que é conhecidamente mais rápido que os AG's, a aplicação do movimento Troca Jogos foi realizada de forma natural, sem perda de desempenho e sem grande exigência computacional. Com isso, o espaço de busca trabalhado pelo *VNS-CS* foi maior, permitindo muitas vezes a obtenção de soluções não alcançadas pelas outras duas abordagens. Em relação aos trabalhos da literatura, o *VNS-CS* conseguiu superar os resultados de duas instâncias, CIRC12 e CIRC14, conforme apresentado na Tabela 6.20. Nas demais instâncias, os resultados foram bem próximos, com *gap* máximo de 5,68%. Além disso, quando comparados os tempos computacionais, as abordagens apresentadas demonstraram um desempenho melhor que as da literatura, que em alguns casos chegam a dias de processamento para encontrar algumas soluções.

O trabalho apresentado pode ser justificado por diversos motivos. Em primeiro lugar o fato de que as escalas de jogos são, na grande maioria dos casos, geradas de forma manual, o que demanda muito tempo e resulta em tabelas que nem sempre conseguem atender a todos os requisitos, além de não considerarem a rota de viagem dos times no decorrer da competição. Outro motivo que releva o trabalho desenvolvido é que, embora haja na literatura outras abordagens evolutivas aplicadas às variações do TTP, a abordagem proposta neste trabalho inova ao utilizar uma representação genética compacta (cromossomos), junto a qual não há inviabilidade, associada a um algoritmo de expansão de código, que transforma cromossomos em escalas de jogos, sendo estas exploradas pela metaheurística SA. Desta forma, o AG não trabalha diretamente com escalas de jogos, pois esta tarefa é a grande, senão a maior, dificuldade de aplicar algoritmos evolutivos na resolução do TTP.

Além disso, a aplicação de técnicas que realizam busca através de agrupamentos na resolução do mTTP é inédita. Conforme apresentado, estas técnicas podem diminuir o esforço computacional de outras metaheurísticas, pois presume-se que a detecção de possíveis regiões promissoras pode encurtar o caminho a ser percorrido até soluções de alta qualidade.

Enfim, os resultados apresentados demonstram o potencial das técnicas utilizadas nas três abordagens apresentadas neste trabalho, uma vez que para todas as instâncias testadas elas conseguiram chegar a resultados muito próximos aos melhores conhecidos da literatura e, em muitos casos, com tempo computacional inferior.

Como possíveis extensões deste trabalho sugere-se, em primeiro lugar, um estudo mais detalhado da técnica *CS, testando outras metaheurísticas como componente ME, pois conforme apresentado, esta abordagem foi superior às demais. Outra questão que é interessante de se testar diz respeito a métrica de distância entre soluções. Novas métricas associadas a graus de similaridade entre soluções, diferentes dos usados neste trabalho, são tópicos importantes de serem estudados, além de permitirem comparações entre as metodologias. A implementação de novas técnicas de assimilação também pode gerar resultados interessantes, permitindo explorações diferentes das trajetórias que conectam duas soluções. Por fim, adaptar os algoritmos apresentados para que os mesmos abordem restrições que representem problemas reais, como por exemplo, o problema do Campeonato Brasileiro de Futebol. Este vem sendo estudado por alguns autores e por isso já possui uma biblioteca de instâncias disponíveis para testes, além de possibilitar comparações mais realísticas com as escalas oficiais.

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. 29, 37, 40, 41

ARMSTRONG, J.; WILLIS, R. J. Scheduling the cricket world cup: A case study. **Journal of the Operational Research Society**, v. 44, p. 1067–1072, 1993. 38

BEAN, J. C.; BIRGE, J. R. Reducting travelling costs and player fatigue in national basketball association. **Interfaces**, v. 10, n. 3, p. 98–102, 1980. 37

BENOIST, T.; LABURTHE, F.; ROTTEMBOURG, B. Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems. In: **Proceedings**. [S.l.]: Proceedings Of Cpaior'01, 2001. 29

BIAJOLI, F. L.; CHAVES, A. A.; MINE, O. M.; SOUZA, M. J. F. Escala de jogos de torneios esportivos: Uma abordagem via simulated annealing. In: XXXV SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 35. **Anais**. Natal - RN, 2003. p. 1295–1306. 36, 38, 39

_____. **Resolução do Problema de Programação de Jogos do Campeonato Brasileiro de Futebol**. Ouro Preto - MG, 2003. 60 p. Relatório técnico. Disponível em: <http://www.decom.ufop.br/prof/marcone/Orientacoes/PJCBviaSimulatedAnnealing.pdf>. Acesso em: 23 Jan. 2007. 29, 30, 35, 36, 38, 39

BIAJOLI, F. L.; CHAVES, A. A.; MINE, O. M.; SOUZA, M. J. F.; PONTES, R. C.; LUCENA, A.; CABRAL, L. F. Scheduling the brazilian soccer championship: A simulated annealing approach. In: FIFTH INTERNATIONAL CONFERENCE ON THE PRACTICE AND THEORY OF AUTOMATED TIMETABLING (PATAT 2004), 5. **Proceedings**. Pittsburgh, USA, 2004. p. 433–437. 36, 38, 39

BIAJOLI, F. L.; LORENA, L. A. N. Uma abordagem evolutiva para o mirrored traveling tournament problem. **IV WORCAP - Workshop dos Cursos de Computação Aplicada do INPE**, São José dos Campos - SP, 2005. 67

_____. Mirrored traveling tournament problem: An evolutionary approach. **Springer Lecture Notes in Artificial Intelligence Series**, v. 4140, p. 208 – 217, 2006. 67

CHAVES, A. A.; LORENA, L. A. N. Hybrid algorithms with detection of promising areas for the prize collecting travelling salesman problem. In: FIFTH

INTERNATIONAL CONFERENCE ON HYBRID INTELLIGENT SYSTEMS (HIS'05). **Proceedings**. Rio de Janeiro, RJ, 2005. p. 49–54. 57

CONCíLIO, R. **Contribuições à solução de problemas de escalonamento pela aplicação Conjunta de computação evolutiva e otimização com restrições**. 118 p. Dissertação (Mestrado) — Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação - UNICAMP, Campinas - SP, 2000. 37, 38

CONCíLIO, R.; ZUBEN, F. J. Uma abordagem evolutiva para geração automática de turnos completos em torneios. **Revista Controle e Automação**, v. 13, n. 2, p. 105–122, 2002. 28, 38

COSTA, D. An evolutionary tabu search algorithm and the nhl scheduling problem. **Infor**, v. 33, n. 3, p. 161–178, 1995. 38, 63

DINITZ, J.; LAMKEN, E.; WALLIS, W. Scheduling a tournament. In: _____. **Handbook of Combinatorial Designs**. [S.l.]: CRC Press, 1995. p. 578–584. 64

DORIGO, M.; MANIEZZO, V.; COLORNI, A. The ant system: Optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics-Part B**, v. 26, n. 1, p. 29–41, 1996. 29

DOWSLAND, K. A. Simulated annealing. In: _____. **Modern Heuristic Techniques for Combinatorial Problems**. Blackwell Scientific Publications, London: Advanced Topics in Computer Science Series, 1993. cap. 2, p. 20–69. 29

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), 7. [S.l.]: Proceedings, 2001. p. 580–589. 29, 30, 33, 37, 38

_____. Sports scheduling. In: _____. **Handbook of Scheduling: Algorithms, Models and Performance Analysis**. [S.l.]: CRC Press, 2004. cap. 52, p. 1–19. 38

EIBEN, A. E.; RUTTKAY, Z. Constraint-handling techniques. In: _____. **Handbook of Evolutionary Computation**. Oxford: University Press, 1997. 67

ELF, M.; JüNGER, M.; RINALDI, G. Minimizing breaks by maximizing cuts. **Operations Research Letters**, v. 31, p. 343–349, 2003. 41

ELMOHAMED, S.; CODDINGTON, P.; FOX, G. A comparison of annealing techniques for academic course scheduling. **Lecture Notes in Computer Science**, v. 1408, p. 92–114, 1998. 64

FEO, T.; RESENDE, M. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, p. 109–133, 1995. 29

GLOVER, F. Tabu search: Part 1. **ORSA Journal of Computing**, v. 1, p. 190–206, 1989. 29

_____. Tabu search: Part 2. **ORSA Journal of Computing**, v. 2, p. 4–32, 1990. 29

_____. New ejection chain and alternating path methods for traveling salesmas problems. **Computer Science and Operations Research: New Developments in Their Interfaces**, p. 449–509, 1992. 62

_____. Ejection chains: Reference structures and alternating path methods for traveling salesman problems. **Discrete Applied Mathematics**, v. 65, p. 223–253, 1996. 62

_____. Tabu search and adaptive memory programing: Advances, applications and challenges. In: _____. **Interfaces in Computer Science and Operations Research**. [S.l.]: Kluwer, 1996. p. 1–75. 46

GLOVER, F.; LAGUNA, M. Tabu search. In: _____. **Modern Heuristic Techniques for Combinatorial Problems, Advanced Topics in Computer Science Series**. London: Blackwell Scientific Publications, 1993. cap. 3, p. 70–150. 29

GOLDBERG, D. E. **Genetic Algorithms in Search. Optimization and Machine Learning**. Addison-Wesley: Berkeley, 1989. 223 p. 29

HAMIEZ, J. P. E.; HAO, J. K. Solving the sports league scheduling problem with tabu search. **Lecture Notes In Artificial Intelligence**, v. 2148, p. 24–36, 2001. 37, 39

HENTENRYCK, P. V.; VERGADOS, Y. Traveling tournament scheduling: A systematic evaluation of simulated annealling. In: **CPAIOR**. [S.l.: s.n.], 2006. p. 228–243. 21, 41, 82, 91, 92, 93

HENZ, M. Scheduling a major college basketball conference: Revisited. **Operations Research**, v. 49, n. 1, 2001. 38

HENZ, M.; MULLER, T.; THIEL, S. Global constraints for round robin tournament scheduling. **European Journal of Operational Research**, v. 153, p. 92–101, 2004. 29

HOLLAND, J. H. **Adaptation in natural and artificial systems**. Ann Arbor, 1975. 211 p. 49

KIRKPATRICK, S.; GELLAT, D.; VECCHI, M. Optimization by simulated annealing. **Science**, v. 220, p. 671–680, 1983. 29, 48

LOURENçO, H. R.; MARTIN, O.; STüTZLE, T. Iterated local search. In: _____. **Handbook of Metaheuristics**. Norwell: Kluwer Academic Publishers, 2002. p. 321–353. 52

MCALOON, K.; TRETKOFF, C.; WETZEL, G. Sports league scheduling. In: **Proceedings of the Third ILOG Optimization Suite International Users' Conference**. Paris, France: Proceedings, 1997. 39

MINE, M. T.; SILVA, M. S. A.; SOUZA, M. J. F.; SILVA, G. P.; OCHI, L. S. Busca local iterada aplicada à resolução do problema de programação de jogos de competições esportivas ralizadas em dois turnos espelhados. In: V ENCONTRO REGIONAL DE MATEMáTICA APLICADA E COMPUTACIONAL, 35. **Anais do ERMAC 2005**. Natal - RN, 2005. v. 1, p. 1–6. 36

MIYASHIRO, R.; IWASAKI, H.; MATSUI, T. Characterizing feasible pattern sets with a minimun number of breaks. In: FOURTH INTERNATIONAL CONFERENCE ON THE PRACTICE AND THEORY OF AUTOMATED TIMETABLING (PATAT2002). [S.l.]: Proceedings, 2002. p. 311–313. 29, 37, 41

MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. **Computers and Operations Research**, v. 24, p. 1097–1100, 1997. 29, 46, 50

NEMHAUSER, G.; TRICK, M. A. Scheduling a major college basketball conference. **Operations Research**, v. 46, n. 1, p. 1–8, 1998. 38

OLIVEIRA, A. C. M. **Algoritmos Evolutivos Híbridos com Detecção de Regiões Promissoras em Espaços de Busca Contínuo e Discreto**. 200 p. Tese (Doutorado) — Instituto Nacional de Pesquisa Operacional (INPE), São José dos Campos - SP, 2004. 29, 44, 53, 54, 57, 74

OLIVEIRA, A. C. M.; LORENA, L. A. N. Hybrid evolutionary algorithms and clustering search. **Springer SCI Series**, 2006. 78

_____. Pattern sequencing problems by clustering search. **Springer Lecture Notes in Artificial Intelligence Series**, v. 4140, p. 218 – 227, 2006. 57, 78

RIBEIRO, C. C. Metaheuristics and applications. In: ADVANCED SCHOOL ON ARTIFICIAL INTELLIGENCE. Estoril, Portugal, 1996. 29, 47

RIBEIRO, C. C.; URRUTIA, S. Heuristics for the mirrored traveling tournament problem. In: FIFTH INTERNATIONAL CONFERENCE ON THE PRACTICE AND THEORY OF AUTOMATED TIMETABLING (PATAT2004). Pittsburgh, USA: Proceedings, 2004. p. 423–443. 21, 29, 30, 34, 40, 64, 66, 82, 90, 91, 92

_____. Maximizing breaks and bounding solutions to the mirrored traveling tournament problem. **Discrete Applied Mathematics**, v. 154, p. 1932–1938, 2006. 41, 82

_____. Scheduling the brazilian soccer championship. In: SIXTH INTERNATIONAL CONFERENCE ON THE PRACTICE AND THEORY OF AUTOMATED TIMETABLING (PATAT 2006). **Proceedings**. [S.l.], 2006. p. 481–483. 36

SOUZA, M. J. F. **Notas de aula da disciplina Inteligência Computacional para Otimização**. 2005. Departamento de Ciência da Computação - Universidade Federal de Ouro Preto. Disponível em: <http://www.decom.ufop.br/prof/marcone>. Acesso em: 22 Nov. 2006. 46, 49, 51, 52

SYSWERDA, G. Uniform crossover in genetic algorithms. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS(ICGA). Virginia, USA: Proceedings, 1989. p. 2–9. 68

TOURN. **Challenge Traveling Tournament Instances**. 2007. Michael Trick's Website. Referência on-line. Disponível em: <http://mat.gsia.cmu.edu/TOURN/>. Acesso em: 22 jan. 2007. 34, 81, 82

TRICK, M. A. A schedule-then-break approach to sports timetabling. **Lecture Notes In Computer Science**, v. 2079, p. 242–252, 2001. 38

WERRA, D. D. Geography, games and graphs. **Discrete Applied Mathematics 2**, p. 327–337, 1980. 41

WHITLEY, D.; GORDON, V. S.; K., M. Lamarckian evolution, the baldwin effect and function optimization. In: THIRD CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE. Berlin, Springer: Proceedings, 1994. p. 6–15. 50, 67

WILLIS, R. J.; TERRIL, B. J. Scheduling the australian state cricket season using simulated annealing. **Journal of the Operational Research Society**, v. 45, p. 276–280, 1994. 38

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)**. Honolulu: Proceedings, 2002. p. 1660–1665. 38, 39

ZHANG, H. Generating college conference basketball schedules by a sat solver. In: FIFTH INTERNATIONAL SYMPOSIUM ON THE THEORY AND APPLICATIONS OF SATISFIABILITY TESTING (SAT 2002). Cincinnati: Proceedings, 2002. p. 281–291. 29

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

A PARÂMETROS UTILIZADOS NOS ALGORITMOS

Tabela A.1 - Parâmetros utilizados no AG

Algoritmo Genético	
Tamanho da População Inicial	300
Nº de Indivíduos novos a cada geração	100
Nº de gerações	10
Probabilidade de Mutação	30%

Tabela A.2 - Parâmetros utilizados no algoritmo *SA*

Simulated Annealing	
Temperatura Inicial	1000000
Temperatura de Resfriamento	0.01
Nº de Iterações / Valor de Temperatura	3000
Índice de Resfriamento	5%

Tabela A.3 - Parâmetros utilizados no algoritmo *ILS*

Iterated Local Search	
Nº máximo de iterações do ILS ($IterMax$)	1000
Nº de reinicializações (NR)	3
Nº máximo de iterações sem melhora ($IterSM$)	50

Tabela A.4 - Parâmetros utilizados no algoritmo ECS

Evolutionary Clustering Search	
Qtde máxima de *cluters* (NT_c)	20
Pressão de densidade do AA (PD)	2
Nº de soluções selecionadas a cada geração (NS)	200

Tabela A.5 - Parâmetros utilizados no algoritmo *VNS-CS*

VNS-Clustering Search	
Nº máximo de iterações do VNS ($IterMax$)	1000
Qtde máxima de *cluters* (NT_c)	20
Pressão de densidade do AA (PD)	2
Nº de soluções selecionadas a cada geração (NS)	300

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Propostas e Relatórios de Proje-tos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanha-mento de projetos, atividades e convê- nios.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Relatórios de Pesquisa (RPQ)

Os Relatórios de Pesquisa reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Publicações Didáticas (PUD)

As Publicações Didáticas incluem apostilas, notas de aula e manuais didáticos.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou inter- pretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto os executáveis.