CAPÍTULO 5

CASO 1: GERAÇÃO DE TURNOS COMPLETOS EM TORNEIOS

5.1 Motivação Inicial

Como motivação inicial para o encaminhamento desta dissertação, abordou-se o caso de geração automática de turnos completos em torneios. Esse estudo de caso apresenta-se com características análogas às presentes no problema da produção da grade horária, sendo que ambos possuem múltiplos objetivos e múltiplas restrições, embora as condições de contorno impostas sejam outras (peculiaridades de cada problema). Apresenta, ainda, uma explosão combinatória de candidatos à solução, de modo que uma busca exaustiva pela solução ótima representaria um procedimento computacionalmente intratável.

5.2 Formulação do Problema

Uma das principais contribuições deste trabalho está na solução do problema de representação e na produção de soluções factíveis para o caso específico de geração de turnos completos em torneios de competição, baseados em partidas entre dois participantes adversários (CONCILIO & VON ZUBEN, 2000a; CONCILIO & VON ZUBEN, 2000b), tais como campeonatos esportivos, maratonas, competições escolares, etc. As principais restrições a serem atendidas são:

- cada participante deve enfrentar todos os seus adversários uma única vez durante o torneio;
- todos os *n* participantes, a menos no caso de constituírem um número ímpar, devem jogar a cada rodada.

Portanto, o objetivo é elaborar uma tabela de jogos completa, formando um turno com *n*-1 rodadas, sendo que esta tabela será evoluída geração a geração pela aplicação de operadores genéticos devidamente adaptados ao contexto. O problema apresentado acima mostra-se compatível com o tipo de solução a ser proposta, uma vez que é fácil perceber a

explosão combinatória existente. Em outras palavras, existirá uma grande quantidade de combinações para a formação dos jogos, sendo que este valor será função do número de participantes. Dessa maneira, o algoritmo a ser proposto realizará uma busca entre os possíveis candidatos à solução ótima (ou aproximadamente ótima).

A Tabela 5.1 mostra o número de combinações possíveis para os jogos, de modo a compor um turno completo. Nesse caso, assume-se haver distinção de mando de jogo. Caso o torneio tenha turno e returno, a solução não muda, pois basta produzir o turno, já que o returno seria dado pela inversão simples de mando. O número de turnos completos factíveis para n participantes é dado pela seguinte expressão: $(n-1)!.(n-3)!.(n-5)!.....(n-(n-1))!.\ 2^{(n-1)\times\frac{n}{2}}$ (veja Anexo C). Esta expressão fornece o número de candidatos presentes no espaço de busca da representação expandida.

Tabela 5.1: Número de combinações possíveis para os jogos na composição de um turno completo.

Número de participantes	**Número de combinações**
2	2
4	384
6	$2{,}36\times10^{7}$
8	$9{,}7410\times10^{14}$
10	$4{,}6331\times10^{25}$
12	$3{,}8785\times10^{39}$
14	$8{,}1039\times10^{56}$
16	$5{,}6893\times10^{77}$
18	$1{,}7383\times10^{102}$
20	$2{,}9062\times10^{130}$

O problema de representação compacta foi resolvido a partir da proposição de uma codificação genética original, que utiliza um algoritmo de geração de todas as rodadas (atendendo às duas restrições citadas) apenas a partir da primeira rodada. Portanto, a codificação genética envolve apenas a descrição da primeira rodada. Dessa maneira, a Tabela 5.2 mostra os valores calculados para a definição do número de candidatos factíveis para essa representação genética compacta. Salienta-se que o que importa na primeira rodada são os participantes que a comporão e qual a ordem dos competidores nos pares para a definição dos jogos, indicando quem está no seu domínio ou fora dele. Outro ponto a

ser considerado é que não interessa qual a seqüência de jogos dos pares, tanto na primeira rodada como nas demais.

No exemplo a seguir, para $n = 6$, existirão 6! permutações possíveis, sendo 3! rodadas equivalentes para cada tripla de jogos. Especificamente para o caso dos jogos envolverem os competidores 2 4, 1 6 e 5 3, todas as seguintes rodadas serão equivalentes:

2 4 1 6 5 3
2 4 5 3 1 6
1 6 2 4 5 3
1 6 5 3 2 4
5 3 2 4 1 6
5 3 1 6 2 4

Essas considerações levam à conclusão de que o número de diferentes primeiras rodadas factíveis vai ser dado por: $\frac{n!}{\left(\frac{n}{2}\right)!}$, sendo esta a cardinalidade do espaço de busca na representação compacta.

Tabela 5.2: Número de combinações possíveis para os jogos na composição de uma rodada.

Número de participantes	**Número de combinações**
2	2
4	12
6	120
8	1680
10	$3{,}024\times10^{4}$
12	$6{,}6528\times10^{5}$
14	$1{,}7297\times10^{7}$
16	$5{,}1891\times10^{8}$
18	$1{,}7643\times10^{10}$
20	$6{,}7044\times10^{11}$

Por outro lado, o problema de geração de soluções factíveis pela aplicação de operadores genéticos junto às soluções existentes se dá pela implementação de sofisticados mecanismos de reparação, com convergência garantida.

Dado o número *n* de participantes, serão necessárias *n*–1 rodadas para que as restrições impostas sejam atendidas, de modo que um candidato à solução do problema será formado pelo conjunto das *n*–1 rodadas que satisfazem as condições especificadas.

O *fitness* será calculado pela soma ponderada de dois termos. O primeiro termo da função (FC_1) contabiliza o número de participantes do torneio que realizam *r* jogos consecutivos dentro (ou fora) de seu domínio. O valor de *r* é função de *n*, sendo que nenhum participante pode realizar mais que *r* jogos consecutivos dentro (ou fora) de seu domínio, pois existe uma restrição que impede esta ocorrência. O segundo termo da função-objetivo (FC_2) é uma medida da diferença entre as distâncias máxima e mínima percorridas pelos participantes do torneio considerando todas as rodadas. É desejável que, para completar o torneio, todos os participantes tenham percorrido uma distância parecida, de modo a não privilegiar este ou aquele participante em termos de custo e tempo de deslocamento. Vale lembrar que cada jogo sempre ocorre no domínio de um dos dois participantes.

Dessa maneira, o problema de otimização apresentado pode ser definido como a seguir, sendo dados os valores de *n* e *r*, e assumindo que $x \in CR_n$ (CR_n representa todos os possíveis conjuntos de *n*–1 rodadas):

$$\max_x \quad w_1 \cdot \frac{1}{FC_1(x)} + w_2 \cdot \frac{1}{FC_2(x)}$$

sujeito a:

- $w_1, w_2 > 0$ e $w_1 + w_2 = 1$;
- cada um dos participantes joga contra todos os outros;
- cada participante joga uma única vez em cada rodada;
- para cada participante, a diferença do número de jogos em seu domínio e fora dele é no máximo um;
- número máximo de jogos consecutivos no domínio do participante (ou fora) não é superior a *r*.

5.3 Exemplo ilustrativo

Considere o seguinte exemplo para cálculo de FC_1 e FC_2, adotando $n = 8$, $r = 3$ e $w_1 = w_2 = 0{,}5$. A matriz D contém as distâncias entre o domínio de cada participante em relação ao domínio dos demais.

$$D = \begin{bmatrix} 0 & 0 & 0 & 435 & 435 & 1123 & 1123 & 2783 \\ 0 & 0 & 0 & 435 & 435 & 1123 & 1123 & 2783 \\ 0 & 0 & 0 & 435 & 435 & 1123 & 1123 & 2783 \\ 435 & 435 & 435 & 0 & 0 & 1558 & 1558 & 2491 \\ 435 & 435 & 435 & 0 & 0 & 1558 & 1558 & 2491 \\ 1123 & 1123 & 1123 & 1558 & 1558 & 0 & 0 & 3906 \\ 1123 & 1123 & 1123 & 1558 & 1558 & 0 & 0 & 3906 \\ 2783 & 2783 & 2783 & 2491 & 2491 & 3906 & 3906 & 0 \end{bmatrix}$$

Rodada 1: 4 7 2 6 5 8 1 3
Rodada 2: 6 4 8 3 5 1 7 2
Rodada 3: 2 1 8 6 7 5 3 4
Rodada 4: 4 2 3 7 6 5 1 8
Rodada 5: 8 2 7 6 5 3 1 4
Rodada 6: 3 6 4 8 7 1 2 5
Rodada 7: 6 1 8 7 5 4 2 3

Figura 5.1: Exemplo de candidato factível à solução.

Tabela 5.3: Análise do candidato factível apresentado na Figura 5.1em relação ao *fitness*.

Participante	**Penalidade em relação a *r***	**Distância percorrida**
1	0	2681
2	0	4341
3	0	3218
4	0	2428
5	0	3551
6	0	6152
7	0	6587
8	0	7765
	Somatória= 0	Menor= 2428 e Maior= 7765

O *fitness* é então obtido na forma:

$$\frac{1}{FC_1} = \frac{1}{Somatória + 1} = 1 \qquad \frac{1}{FC_2} = \frac{1}{\frac{Maior}{Menor}} = 0{,}313$$

$$Fitness = w_1 \frac{1}{FC_1} + w_2 \frac{1}{FC_2}$$

$$Fitness = 0{,}5 + 0{,}157 = 0{,}657$$

A penalidade adotada para *r* neste exemplo será sempre acrescida de uma unidade quando, no candidato factível à solução, qualquer participante jogar três ou mais vezes consecutivas dentro ou fora do seu domínio. No candidato à solução apresentado na Figura 5.1 esse fato não ocorre, portanto a penalidade em relação a *r* na Tabela 5.3 será zero.

5.4 Estratégia de Solução

A estratégia de solução permite a elaboração de uma tabela para um número arbitrário de participantes. Outra característica é o fato de que a quantidade de participantes deve ser sempre um número par. Se esse número for ímpar, define-se um competidor fictício, de modo que seu adversário será o participante do torneio que irá folgar naquela rodada.

Para o cálculo do número de jogos necessários, utiliza-se a combinação de *n* participantes dois a dois. Dessa maneira, aplica-se:

$$C_{n,2} = \frac{n!}{(n-2)!2!}.$$

Para exemplificar, tomando o número de participantes igual a 20, resulta:

$$C_{20,2} = \frac{20!}{(20-2)!\,2!} = 190 \text{ jogos}.$$

Como, por hipótese, todos os participantes jogam em todas as $n-1$ rodadas, cada uma delas terá $\frac{n}{2}$ jogos. Para o caso de 20 participantes, serão necessárias 19 rodadas para que todos os jogos sejam realizados, totalizando 190 jogos.

5.4.1 Codificação Compacta e Expansão de Código

A codificação genética utilizada para a composição de um cromossomo em representação compacta (será composto apenas pela primeira rodada), é uma permutação de números entre 1 e *n*. Cada um desses números (genes) representa um participante dentro da tabela gerada. Ressalta-se que essa formação cromossômica compacta vai atender ao elenco de restrições associado a uma rodada específica.

Para a geração da segunda rodada em diante, ou seja, o fenótipo do indivíduo, aplica-se um algoritmo de expansão de código. Para tanto, escolhe-se uma semente do processo de geração aleatória, de modo que todos os outros pares de genes (jogos) sejam produzidos a partir dela. O gerador pseudo-aleatório apresenta a propriedade de repetitividade. Percebe-se, portanto, que a cada primeira rodada gerada está atrelada uma semente, a partir da qual serão codificadas de forma única as demais rodadas. Assim, dada uma primeira rodada e uma semente, o algoritmo de geração da tabela completa vai sempre produzir um mesmo turno, representando o fenótipo associado ao genótipo em representação compacta. É por isso que é dito que a representação genética é compacta, pois a cada conjunto de primeira rodada e semente existe sempre um único turno completo, que vai representar uma solução-candidata factível.

Por exemplo, tomando $n = 16$, começa-se com uma primeira rodada correspondente a uma permutação arbitrária dos números entre 1 e 16, tal como:

16 1 9 11 7 15 4 2 6 10 3 14 12 13 5 8

que dará origem ao seguinte código genético:

16	1	9	11	7	15	4	2	6	10	3	14	12	13	5	8

Esta lista de 16 genes, cada um com um valor distinto, representa um cromossomo, o qual será a entrada para o procedimento de expansão de código, responsável pela geração das demais 14 rodadas. Para tanto, o procedimento de expansão de código irá empregar um gerador de números pseudo-aleatórios que apresente a propriedade de repetitividade, e cada cromossomo contará com um gene adicional que representará a semente da geração (Figura 5.2). Para um mesmo cromossomo e uma mesma semente, sempre irá resultar uma mesma seqüência de rodadas.

16	1	9	11	7	15	4	2	6	10	3	14	12	13	5	8	semente

Figura 5.2: Representação genética compacta (genes e semente).

Após a elaboração da primeira rodada, a produção da seqüência de jogos que comporão as $n-2$ rodadas restantes será realizada com o auxílio de um algoritmo de reparação e outro de busca local. A rotina de reparação será responsável pela aceitação ou rejeição de cada um dos números aleatórios gerados, correspondente a cada gene (participante do torneio).

Tomando como exemplo $n = 16$ e assumindo que a primeira rodada é a apresentada anteriormente, se o gerador de números aleatórios produzir 11 como uma primeira saída, a conseqüência será uma segunda rodada com a seguinte configuração inicial:

11 _ _ _ _ _ _ _ _ _ _ _ _ _ _.

Nesse caso, o único valor que não poderá ser aceito como próxima saída do gerador será 9, pois na primeira rodada já existe o jogo entre os participantes 11 e 9, embora com o domínio invertido.

Com o aumento do número de jogos já definidos, a determinação de uma rodada válida pode se tornar mais custosa computacionalmente, pois aumenta a probabilidade de que a saída do gerador pseudo-aleatório não seja aceita. Este processo de montagem do turno completo a partir da primeira rodada, atendendo a múltiplas restrições, pode ser visto como um processo de busca em uma árvore de decisão, sendo necessário retornar a nós anteriores dessa árvore (*backtracking*) caso não exista possibilidade de progresso a partir das condições existentes. Além desses procedimentos de reparação, mecanismos de busca local são aplicados imediatamente após a conclusão do processo de expansão de código, visando aumentar o *fitness* do indivíduo gerado.

Toda vez que se atinge um folha desta árvore de decisão, a partir do nó raiz, obtém-se uma solução factível. Para um dado gerador de números pseudo-aleatórios, mesmo que seja possível adotar qualquer uma das sementes para iniciar a geração, a propriedade de repetitividade e a conseqüente periodicidade do gerador (veja Anexo A) impedem que exista a possibilidade de se percorrer qualquer ramo factível da árvore de decisão, no caso de haver mais de uma opção. Para que qualquer folha da árvore de decisão, representando

uma solução factível, possa ser obtida como resultado da busca, haveria a necessidade de se recorrer a um elenco de geradores de números pseudo-aleatórios, todos eles apresentando a propriedade de repetitividade. Cada expansão de código seria então realizada por um único gerador do elenco de geradores. Assim, podendo variar a semente e também o gerador, seria possível garantir a completitude do processo de expansão de código.

5.4.1.1 Algoritmo de Expansão de Código para Produção de Soluções Factíveis

Para a codificação compacta, a partir de um conjunto de primeira rodada e semente, será elaborado o restante das rodadas pelo emprego de um algoritmo de expansão de código. Salienta-se, novamente, que todas as rodadas devem sempre atender às restrições apresentadas para a sua formação. Se não forem observadas estas restrições, certamente será gerada uma solução infactível. Dessa maneira, lança-se mão de um algoritmo de expansão de código, basicamente composto dos seguintes passos:

- verificar se o jogo (par de participantes) sorteado, ou o seu inverso, já não faz parte de alguma rodada da tabela;
- se não fizer parte, validar esse par e proceder com o sorteio de outro jogo. Caso já esteja presente em alguma rodada, rejeitar o par e realizar outro sorteio para ocupar essa mesma posição.
- este processo se repete até que sejam atendidas as restrições. Caso se esgotem as possibilidades sem levar ao atendimento das restrições, todos os jogos dessa rodada serão invalidados e uma nova seqüência será sorteada;
- outra situação verificada pelo algoritmo de reparação é o número de vezes que uma mesma rodada é invalidada. Se esse valor exceder um determinado número, todas as rodadas (a menos da primeira) são rejeitadas e todo o processo se repete a partir da segunda rodada.

Como a árvore de decisão possui um número finito de nós, é certo que o algoritmo de reparação vai chegar a uma solução factível, embora não se tenha controle sobre o custo computacional específico associado a cada vez que o algoritmo é aplicado. A convergência

ocorreu em todos os casos simulados, para várias instâncias de número total *n* de participantes.

O algoritmo de expansão de código representa uma etapa fundamental na geração de soluções factíveis a partir de uma representação compacta, sendo aplicado toda vez que a representação compacta sofrer qualquer tipo de modificação. Além disso, no caso da representação expandida, este algoritmo também exerce um papel importante, pois a população inicial, assim como novos indivíduos factíveis que sejam gerados ao longo do processo evolutivo, acabam empregando este mesmo algoritmo para se obter conjuntos de *n*−1 rodadas que sejam factíveis.

5.4.2 Codificação expandida

No caso da codificação expandida, os operadores genéticos vão atuar diretamente sobre o conjunto de *n*−1 rodadas. Para se produzir conjuntos completos de *n*−1 rodadas, o mesmo procedimento de expansão de código adotado na seção anterior será necessário. No entanto, ele somente é aplicado na geração da população inicial e de novos candidatos factíveis a serem evoluídos.

No capítulo 4, todas as principais distinções entre as codificações expandida e compacta foram salientadas. Como aspectos comuns, ambas as codificações fazem uso dos mesmos mecanismos de reparação e busca local, os quais serão detalhados a seguir.

5.4.3 Algoritmo de Busca Local

O processo de busca local é aplicado a todos os indivíduos, sejam eles gerados aleatoriamente para a composição da população ou após a aplicação dos operadores genéticos. O mecanismo é o mesmo tanto para representação compacta como para representação expandida.

A busca local permite a geração de candidatos factíveis e a produção de ótimos locais. Em outras palavras, desde que não haja infactibilidade da solução e para a melhoria do valor do *fitness* de um dado conjunto de *n*−1 rodadas, será possível fazer uma inversão do domínio de cada uma das partidas envolvidas. Outra possibilidade existente é a reordenação das rodadas já definidas.

No caso da representação compacta, todas as vezes em que o procedimento de expansão do código for aplicado em conjunto com o algoritmo de busca local, sua execução a partir de um mesmo código genético compacto levará sempre a um mesmo fenótipo.

5.4.4 Operação do Algoritmo Genético

A descrição que se segue também será válida, com exceção de alguns módulos específicos, tanto para a representação compacta quanto a expandida. Em linhas gerais, o programa calcula o *fitness* de cada indivíduo de uma população inicial gerada. Essa população é ordenada levando-se em conta o *fitness* obtido para, a seguir, receber a aplicação do algoritmo de *Roulette Wheel* e, em seqüência, os operadores genéticos. Novamente a população é ordenada para receber uma seleção parcialmente elitista.

Após a escolha dos melhores cromossomos, novos indivíduos serão gerados para a complementação da população. Dessa maneira, a próxima geração está pronta para sofrer os passos já descritos. Ao término de todas as gerações será apresentada a solução que alcançar o melhor resultado de *fitness* segundo os critérios especificados.

A seguir, são brevemente descritos os módulos que compõem o programa computacional que foi implementado.

- **Mostra_Distâncias**: contém as distâncias entre as localidades dos participantes do torneio.
- **Rodadas_Repetidas**: faz uma busca, verificando se os participantes têm *r* jogos consecutivos dentro do seu domínio (mando de jogo) ou fora.
- **Pop_Inicial**: faz a geração da população inicial.
- **Expansão_Código:** responsável pela geração da segunda rodada em diante (emprega o código genético da primeira rodada mais uma semente).
- **Mostra_Tabelas**: faz a exibição da tabela com todas as rodadas definidas.
- **Mostra_Estatística**: faz a verificação se as rodadas estão distribuídas com equilíbrio entre jogos no domínio e fora, em relação aos mandos de jogos dos participantes.
- **Robin_Hood**: de acordo com os resultados de saída do módulo **Mostra_Estatística**, faz a inversão automática de alguns mandos de jogos para equilibrar a tabela.

- **Verifica_Mando**: faz uma verificação dos mandos de jogo dos participantes. Caso haja *r* consecutivos, a solução candidata sofrerá uma punição (degradação do *fitness*).
- **Verifica_Fora**: faz uma verificação do número de jogos fora do domínio dos participantes. Caso haja *r* consecutivos, a solução candidata sofrerá uma punição (degradação do *fitness*).
- **Fit_Mando**: faz o cálculo da segunda parcela do *fitness* de cada indivíduo (solução candidata) da população, levando em consideração os resultados dos módulos **Verifica_Mando** e **Verifica_Fora**.
- **Fit_Distância**: faz o cálculo do *fitness* de cada indivíduo (solução candidata) da população, levando em consideração as distâncias percorridas pelos participantes quando for necessário viajar até o local da partida.
- **Ordenação**: faz a ordenação decrescente em relação ao *fitness* da população.
- **Guarda_Melhor_Indivíduo**: armazena o melhor indivíduo da população para aplicação do processo de seleção salvacionista na produção da próxima geração.
- **Verifica_Melhor_Indivíduo**: o melhor indivíduo da geração anterior será colocado na atual, caso seu *fitness* seja melhor que o de todos os indivíduos da geração atual.
- **Roulette_Wheel**: aplica o algoritmo da roleta para executar o sorteio dos cromossomos-pais que serão utilizados pelos operadores genéticos.
- **Crossover_Ox**: aplica como operador genético o *order crossover* para a geração dos cromossomos-filhos.
- **Mutação_Inversiva**: aplica como operador genético a mutação inversiva, fazendo a troca de posição entre dois genes de um mesmo cromossomo.
- **Seleção_Elitista**: mantém compulsoriamente um terço dos melhores indivíduos da geração atual na nova geração.
- **Guarda_Melhor_Solução**: armazena a melhor solução gerada a partir do indivíduo com o maior *fitness* de todas as gerações.
- **Recupera_Melhor_Tabela**: faz a recuperação da tabela que contém o maior *fitness* de todas as gerações.

5.4.5 Discussão

É importante ressaltar que o problema estudado tem características próprias que indicam o uso da computação evolutiva, mais precisamente, os conceitos de algoritmos genéticos com busca local, também conhecidos na literatura como algoritmos meméticos (MOSCATO, 1989). Essa abordagem depende, no entanto, do atendimento de restrições de diversas naturezas, conforme já discutido.

O algoritmo de reparação é responsável por validar o conjunto de $n-1$ rodadas, tornando a solução candidata factível. Uma vez que as soluções factíveis da geração corrente tenham sido definidas seguindo os critérios mencionados, uma medida da sua qualidade poderá ser obtida, o que equivale a uma formalização matemática do *fitness* de cada indivíduo. Essa medida pode levar em conta mais de um critério, os quais podem ser ponderados de acordo com a importância atribuída a cada um. Em nosso caso, foram estabelecidos dois critérios igualmente importantes, de modo que os dois valores calculados contribuem igualmente para compor a medida do *fitness* de cada indivíduo.

Em relação aos operadores genéticos (*order* c*rossover* e mutação inversiva) é realizado um sorteio pelo algoritmo da roleta para encontrar os cromossomos que serão modificados. Para essa nova população de soluções candidatas, é calculado novamente o *fitness* (seguindo os critérios já abordados). A seguir, aplica-se a ordenação da população (em função do *fitness*), sendo que a partir deste momento ela é submetida ao processo de seleção parcialmente elitista. É mantido um terço da população atual para a próxima geração e todo o processo descrito se repete até que seja alcançada a última geração.

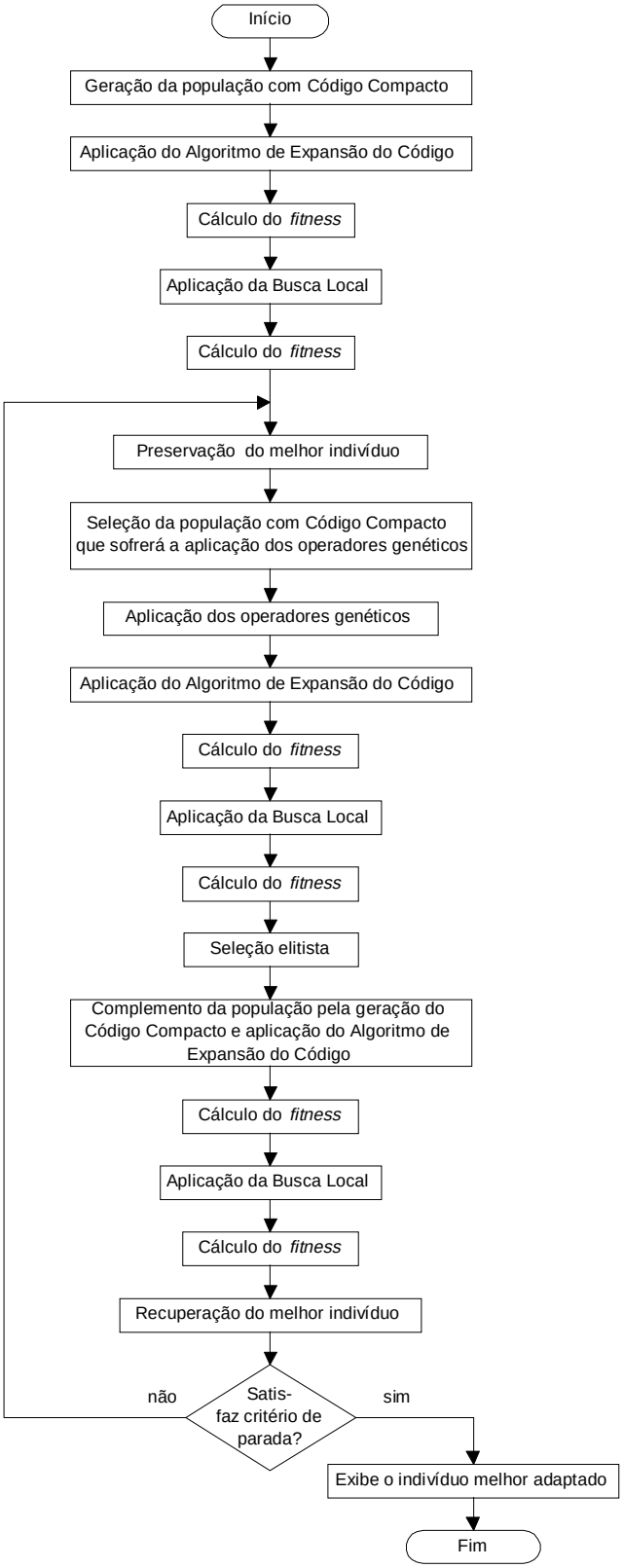
No momento da elaboração da próxima geração, é verificado se o melhor indivíduo da geração anterior está presente na atual. Caso não esteja, ele será introduzido para que não ocorra uma perda do melhor indivíduo e, por conseqüência, uma queda do *fitness*.

Após cada aplicação de operadores genéticos, é ativado o procedimento de busca local. Como resultado final, o programa apresenta a tabela que alcançou o maior valor para o *fitness* e qual é este valor.

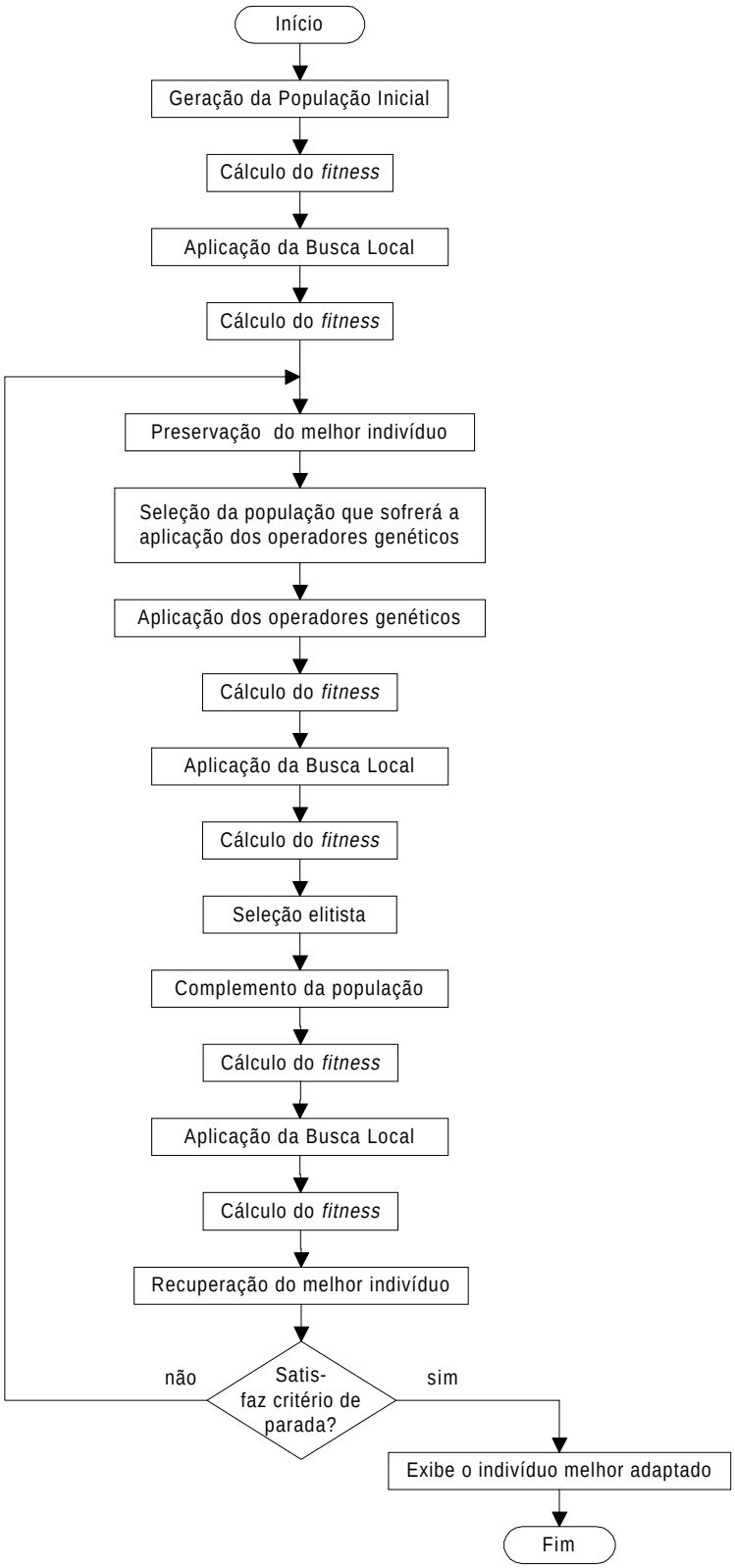
O custo computacional para a geração da população aumenta proporcionalmente aos custos associados ao algoritmo de busca local. Entretanto, resultados muito superiores para o *fitness*, tanto para a codificação compacta quanto para a expandida, foram alcançados

com essa nova metodologia (quando comparados aos valores obtidos sem sua aplicação), por requererem um número reduzido de gerações junto ao processo evolutivo.

5.5 Fluxograma Representativo do Processo Evolutivo com Representação Compacta



5.6 Fluxograma Representativo do Processo Evolutivo com Representação Expandida



5.7 Resultados

Como resultados, apresentam-se na Tabela 5.4 simulações para alguns valores de número de participantes (*n*). O programa computacional foi desenvolvido com a linguagem de programação *Pascal* e os tempos obtidos foram para o processamento em um microcomputador *Pentium* III 500 MHz, com 128Mbytes de memória *RAM*.

Para efeito de comparação com a solução fornecida por um especialista em um problema do mundo real, são apresentados a seguir os resultados obtidos pelo programa desenvolvido neste estudo e a tabela utilizada no Campeonato Paulista de Futebol de 1997, em sua Divisão A1. Utilizou-se a tabela deste ano, porque de 1998 em diante o modo de disputa foi alterado, não atendendo mais à restrição de que cada time deveria jogar contra todos os demais adversários.

Tabela 5.4: Resultados de simulações para diferentes números de participantes (*n*) e diferentes parâmetros.

	Número de participantes	**Número de Gerações**	**Tamanho da população**	**Crossover (%)**	**Mutação (%)**	**Melhor *Fitness***	**Tempo de execução**
1	10	20	40	65	2	0,738	3 min
2	12	20	40	65	2	0,707	19 min
3	12	20	60	65	2	0,717	30 min
4	12	10	40	65	2	0,737	11 min
5	12	20	40	65	4	0,709	13 min
6	12	20	40	80	2	0,718	19 min
7	12	20	40	80	4	0,706	19 min
8	16	20	40	65	2	0,666	1h 08 min
9	16	30	20	65	2	0,629	1h 05 min
10	16	10	30	65	2	0,629	19 min

Nota-se que, ao comparar o *fitness* da Tabela 5.6, elaborada pelo programa, com o da Tabela 5.5, utilizada no Campeonato Paulista de 1997, resultados superiores foram obtidos a partir da aplicação da estratégia proposta neste trabalho. Vale ressaltar que não há conhecimento por parte dos autores deste trabalho acerca do procedimento adotado pelo especialista na produção da Tabela 5.5. Pode ter havido apenas a preocupação da geração de uma solução factível, seguida ou não de etapas de refinamento em busca do atendimento de objetivos, não necessariamente os mesmos adotados neste trabalho. Sendo assim, as duas soluções que estão sendo comparadas podem ter sido obtidas a partir de problemas de otimização com formulação distinta. Como, neste trabalho, o problema foi formulado procurando-se considerar aspectos de grande interesse prático, justifica-se a comparação

realizada com uma solução adotada na prática, independente dos detalhes de implementação desta última.

Tabela 5.5: Solução fornecida pelo especialista (n=16).

***Fitness* 1/FC$_1$= 0,333** ***Fitness* 1/FC$_2$= 0,233**
***Fitness* total= 0,283 (w$_1$=w$_2$=0,5)**
Melhor tabela (solução factível)

Rodada 1=>	1 2	3 4	5 6	7 8	9 10	11 12	13 14	15 16
Rodada 2=>	15 14	4 5	2 8	1 7	6 3	10 11	16 13	12 9
Rodada 3=>	6 4	5 7	8 1	3 2	11 13	14 16	9 15	12 10
Rodada 4=>	4 8	2 5	1 6	7 3	10 14	15 11	16 12	13 9
Rodada 5=>	4 7	5 1	8 3	10 13	11 16	14 9	12 15	6 2
Rodada 6=>	15 10	2 4	5 8	3 1	6 7	11 14	12 13	9 16
Rodada 7=>	3 5	8 6	7 2	1 4	16 10	9 11	14 12	13 15
Rodada 8=>	7 14	2 10	3 12	5 13	4 9	6 15	8 16	1 11
Rodada 9=>	12 2	13 3	9 5	14 4	15 7	16 6	11 8	10 1
Rodada 10=>	2 13	3 9	5 14	7 16	6 11	8 10	1 12	4 15
Rodada 11=>	9 2	14 3	15 5	16 4	11 7	10 6	12 8	13 1
Rodada 12=>	7 10	2 14	3 15	5 16	4 11	6 12	8 13	1 9
Rodada 13=>	15 2	16 3	11 5	10 4	12 7	13 6	9 8	14 1
Rodada 14=>	2 16	3 11	5 10	7 13	6 9	8 14	1 15	4 12
Rodada 15=>	11 2	10 3	12 5	13 4	9 7	14 6	15 8	16 1

Tabela 5.6: Solução fornecida pelo algoritmo genético com representação compacta e busca local (n=16).

número de gerações=20 **tamanho da população=40**
taxa de crossover= 65% **taxa de mutação= 2%**
tempo de execução=1h 08min
***Fitness* 1/FC$_1$= 1,000** ***Fitness* 1/FC$_2$= 0,333**
***Fitness* total= 0,666 (w$_1$=w$_2$=0,5)**
Melhor tabela (solução factível)

Rodada 1=>	16 1	9 11	7 15	4 2	6 10	3 14	12 3	5 8
Rodada 2=>	15 13	10 2	5 9	11 7	6 1	4 16	8 14	3 12
Rodada 3=>	1 3	10 15	9 4	8 16	14 7	12 11	13 6	2 5
Rodada 4=>	6 4	7 8	12 9	11 3	13 14	16 10	15 5	1 2
Rodada 5=>	7 2	14 1	10 13	3 16	8 11	4 12	9 15	5 6
Rodada 6=>	15 8	3 7	5 13	11 16	9 1	4 10	2 14	6 12
Rodada 7=>	11 6	12 5	15 3	13 9	8 2	7 10	16 14	1 4
Rodada 8=>	10 9	13 4	7 6	1 12	16 5	8 3	14 11	2 15
Rodada 9=>	12 2	4 7	9 8	6 16	11 13	5 14	15 1	3 10
Rodada 10=>	7 13	14 9	2 3	4 11	10 12	16 15	5 1	6 8
Rodada 11=>	12 7	1 10	16 9	11 5	14 15	8 4	13 2	3 6
Rodada 12=>	1 8	4 14	9 3	10 11	15 12	2 6	7 5	13 16
Rodada 13=>	3 13	5 4	2 9	12 8	11 1	16 7	14 10	6 15
Rodada 14=>	13 1	5 3	11 2	8 10	7 9	16 12	14 6	4 15
Rodada 15=>	13 8	3 4	12 14	10 5	9 6	15 11	1 7	2 16

Existe uma sensível diferença em relação aos dois valores de penalidade para *r* calculados para as Tabelas 5.5 e 5.6. Para o caso da Tabela 5.6, $\frac{1}{FC_1} = 1{,}000$, indicando que nenhum participante do torneio jogou três vezes consecutivas dentro ou fora do seu domínio. Já para a Tabela 5.5, $\frac{1}{FC_1} = 0{,}333$, o que mostra que houve a participação de alguns competidores em pelo menos três rodadas consecutivas dentro ou fora do seu domínio. Em uma análise mais cuidadosa, pode-se verificar que o participante 3 jogou três vezes consecutivas no seu domínio nas rodadas seis, sete e oito. O participante 10, também nessas mesmas rodadas, fez três partidas consecutivas fora do seu domínio. Essa situação implica que o somatório em relação à *r* será dois, portanto $\frac{1}{FC_1} = \frac{1}{2+1} = 0{,}333$. Caso se procure eliminar diretamente esta seqüência de três jogos consecutivos para os participantes 3 e 10, no domínio e fora dele respectivamente, pela simples troca de mando de jogo, por exemplo, na rodada oito (substituição do jogo 2 10 por 10 2 e do jogo 3 12 por 12 3) resolve-se o problema imediato para os participantes 3 e 10, mas criam-se outros problemas, como:

- a violação da restrição de que cada participante deva ter no máximo a diferença de um entre jogos dentro e fora do domínio;
- com estas trocas, o participante 2 passou a ter três jogos consecutivos fora do seu domínio.

Conclui-se, portanto, que a diferença de desempenho das duas soluções comparadas não pode ser eliminada de forma elementar, fato que conduz à conclusão de que há uma grande diferença qualitativa entre as duas soluções sob comparação.

Analisando-se $\frac{1}{FC_2}$, verifica-se que quanto maior for esta razão, a diferença entre a menor distância percorrida e a maior será um valor menor. Isso implica em dizer que o torneio está mais equilibrado em relação à distância percorrida pelos participantes.

Para o caso da Tabela 5.6, a Figura 5.3 mostra a situação do melhor *fitness* e do *fitness* médio ao longo das gerações. Nota-se na figura apresentada, que já para a primeira geração o melhor *fitness* é maior que o obtido pelo especialista (Tabela 5.5), entretanto este valor ainda será evoluído ao longo das gerações. Este melhor *fitness* obtido já a partir da primeira rodada se deve à adoção de mecanismos de busca local, fazendo com que cada indivíduo que compõe a primeira geração já corresponda a um ótimo local. A distância existente entre o *fitness* médio e aquele produzido pelo melhor indivíduo demonstra a existência de um nível significativo de diversidade na população ao longo das gerações.

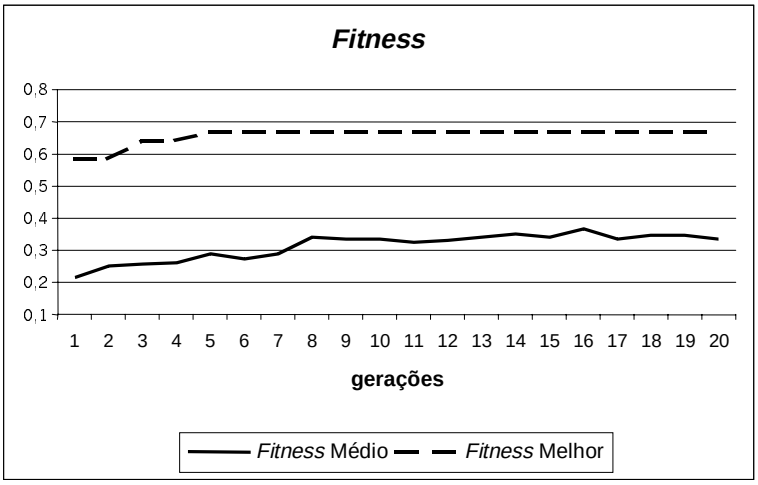


Figura 5.3: Evolução dos *fitness* da média da população e do melhor indivíduo a cada geração, resultando na solução apresentada na Tabela 5.6.

A Tabela 5.7 mostra a média de cinco resultados obtidos em relação ao *fitness* para a codificação genética expandida quando comparada com a compacta. Ressaltando que o termo codificação genética expandida faz referência ao cromossomo que possui no seu genoma todos os genes representativos do torneio. Veja o seguinte exemplo, considerando n=10:

- para a versão compacta, o cromossomo (código genético) terá um tamanho de 10 genes e o seu fenótipo será composto por 90 genes, ou seja, $n \times (n\text{-}1)$;
- para a versão expandida, o cromossomo será composto por 90 genes, sendo que toda esta carga genética será evoluída geração a geração.

Tabela 5.7: Testes comparativos envolvendo a codificação genética compacta e expandida com número de gerações igual a 20 e tamanho da população de 30.

Número de Participantes	**Codificação Expandida**	**Codificação Compacta**
	Fitness médio alcançado	*Fitness* médio alcançado
8	0,598	0,654
10	0,615	0,637
12	0,599	0,607
14	0,612	0,629
16	0,611	0,615
18	0,578	0,600

Analisando-se a tabela anterior, verifica-se que o uso de codificação expandida produz técnicas de busca menos eficientes, quando comparadas à técnica de busca baseada em codificação compacta e expansão de código. A razão para tal é simplesmente o fato de não haver uma evolução de condições iniciais para o algoritmo de otimização baseado em restrições, e pelo fato do processo evolutivo estar operando em um espaço com elementos factíveis e infactíveis, o que impede a aceitação de boa parte das operações genéticas propostas, reduzindo em muito a taxa média de progresso junto ao processo evolutivo.

A próxima tabela mostra o tempo de execução do programa computacional desenvolvido para os mesmos casos da tabela anterior. Os tempos medidos foram obtidos para o processamento em um microcomputador *Pentium* III 650MHz com 128Mbytes de memória *RAM*. O programa computacional foi desenvolvido no ambiente de programação *Delphi*.

Tabela 5.8: Testes comparativos em relação ao tempo de execução do programa com número de gerações igual a 20 e tamanho da população de 30.

Número de participantes	**8**	**10**	**12**	**14**	**16**	**18**
Codificação Compacta	14s	59s	6min 45s	20min 55s	29mim 15s	1h 50min 20s
Codificação Expandida	4s	22s	2min 11s	9min 20s	15min 22s	46min 40s

Nota-se que o tempo de execução para a codificação genética compacta é maior que o da expandida em todos os testes realizados. Justifica-se esse fato pela presença do algoritmo de expansão de código, que faz a geração das demais rodadas do torneio tomando como ponto de partida a primeira rodada e mais uma semente.