CAPÍTULO 4

ABORDAGEM EVOLUTIVA PARA PROBLEMAS COM RESTRIÇÕES

4.1 Considerações Iniciais

Conforme EIBEN & RUTTKAY (1997, C5.7), a aplicação de algoritmos evolutivos na solução de problemas de satisfação de restrições (*constraint-satisfaction problem – CSP*) é interessante sob dois pontos de vista. Por um lado, não se pode esperar que um algoritmo clássico de busca possa alcançar com facilidade a solução de um *CSP*. Determinadas heurísticas de busca apresentam-se bastante eficazes em certos casos, mas falham em outros por restringir o escopo da busca. Ainda segundo EIBEN & RUTTKAY (1997, C5.7), algumas instâncias de *CSPs* podem ser resolvidas pela diversificação da busca, 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 esses princípios são essenciais dentro de algoritmos evolutivos, a sua aplicação para a resolução de *CSPs* apresenta-se promissora.

Por outro lado, algoritmos evolutivos são tradicionalmente utilizados em problemas de otimização que não apresentam nenhum tipo de restrição. Como os operadores convencionais de recombinação e mutação não tratam diretamente restrições, a aplicação de algoritmos evolutivos para problemas com estas características não é imediata. Uma extensão desse fato é a impossibilidade de garantir que pais geneticamente factíveis gerem descendentes também factíveis.

4.2 Definições e Notação Utilizada

A seguir, serão apresentadas algumas definições para posicionar a terminologia utilizada na solução de problemas envolvendo restrições (para maiores detalhes, veja EIBEN & RUTTKAY, 1997, C5.7).

Definição 1: $S = D_1 \times ... \times D_n$ é denominado espaço de busca livre, sendo obtido a partir de um produto cartesiano dos conjuntos D_i, i=1,...,n.

Definição 2: Um problema de otimização sem restrições (*free optimization problem - FOP*) é formado por um par *(S,f)*, sendo *S* um espaço de busca livre e *f* uma função-objetivo em *S*, devendo ser otimizada (minimizada ou maximizada). A solução de um *FOP* é um elemento $s \in S$ com um valor ótimo de *f*.

Definição 3: Um problema de satisfação de restrições (*constraint-satisfaction problem - CSP*) é um par *(S,ϕ)*, sendo *S* um espaço de busca livre e ϕ uma função booleana em *S*. A solução de um *CSP* é um $s \in S$ com $\phi(s)$= verdadeiro.

Definição 4: Um problema de otimização com restrições (*constraint optimization problem - COP*) é uma tripla *(S,f,ϕ)*, sendo *S* um espaço de busca livre, *f* uma função-objetivo em *S* e ϕ uma função booleana em *S*. A solução de um *COP* é um $s \in S$, com um valor ótimo de *f* tal que $\phi(s)$= verdadeiro.

Definição 5: Para o caso de *CSPs* e *COPs*, ϕ será chamada de condição de factibilidade e o conjunto $\{s \in S / \phi(s) =$ verdadeiro$\}$ será o espaço de busca factível.

Por intermédio dessas definições, pode-se representar os problemas *FOP*, *CSP* e *COP* respectivamente pelas seguintes notações: *(S,f,•)*, *(S,•,ϕ)* e *(S,f,ϕ)*, sendo que • implica a ausência do argumento correspondente.

4.3 Transformação de *CSPs* em Problemas que Admitem Solução via Algoritmos Evolutivos

É sabido que a presença de uma função-objetivo (função de *fitness*) para ser otimizada é essencial em algoritmos evolutivos. Pela Definição 3 do item anterior, percebe-se que para um *CSP* (representado pela ênupla $(S,\bullet,\phi)$) isto não ocorre.

Para que um algoritmo evolutivo seja aplicado nesse caso, antes será necessário transformá-lo em um *FOP* - $(S,f,\bullet)$ - ou um *COP* - (S,f,ϕ) -, embora seja sabido que esta transformação nem sempre será possível ou viável.

Definição 6: Considere os problemas P_1 e P_2 como sendo $(S,f,\bullet)$, $(S,\bullet,\phi)$ ou (S,f,ϕ). P_1 e P_2 são equivalentes se:

$$\forall\ s \in S:\ s \text{ é uma solução de } P_1 \Leftrightarrow s \text{ é uma solução de } P_2.$$

Diz-se que P_1 está contido em P_2 se:

$$\forall\ s \in S:\ s \text{ é uma solução de } P_1 \Rightarrow s \text{ é uma solução de } P_2.$$

Assim, a solução de um *CSP* por um algoritmo evolutivo requer que este seja transformado em um *FOP* ou um *COP* que o contém e, em seguida, resolver este novo problema.

Vale o comentário de que *FOPs* permitem uma busca livre, de modo que não apresentam dificuldades para os algoritmos evolutivos. Em contra partida, *COPs* são de difícil solução quando se recorre apenas aos algoritmos evolutivos em sua forma tradicional.

4.4 Espaços de Busca com Restrições

MICHALEWICZ (1997, C5.1) argumenta que muitos dos problemas que apresentam um domínio discreto, tais como *Knapsack problem*, *set covering problem*, *vehicle routing problem* e todos os tipos de escalonamento, *scheduling* e *timetabling*, contêm uma série de restrições que devem ser atendidas simultaneamente.

Acrescenta, ainda, que em geral um espaço de busca *S* consiste de dois subconjuntos disjuntos de subespaços: factíveis (*F* – atendem a todas as restrições simultaneamente) e infactíveis (*U* – violam uma ou mais restrições), sendo que no decorrer do processo de

busca, a população de indivíduos candidatos à solução pode conter elementos que sejam tanto factíveis como infactíveis. A Figura 4.1 ilustra candidatos à solução caracterizados como factíveis, infactíveis e uma solução ótima.

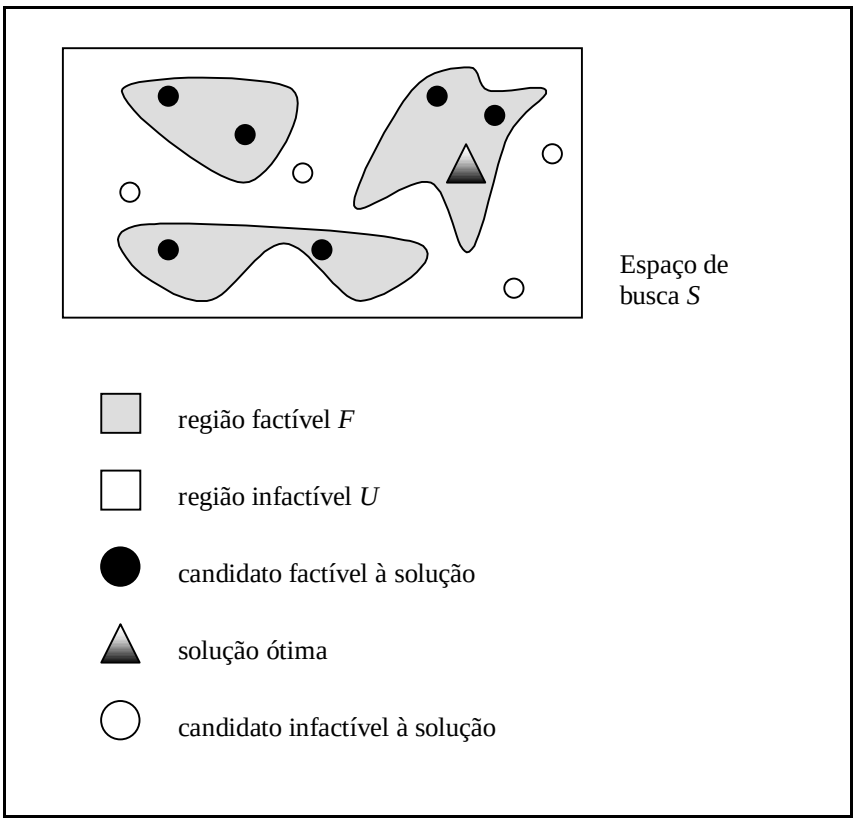

Figura 4.1: Representação pictórica do espaço de busca *S*.

Conforme MICHALEWICZ (1997, C5.1), o problema de como trabalhar com indivíduos infactíveis está longe de ser trivial. De modo geral, duas funções de avaliação devem ser elaboradas: uma delas fará referência ao domínio das soluções factíveis ($eval_f$) e a outra às infactíveis ($eval_u$), tal que:

$$eval_f : F \to \Re \quad \text{e} \quad eval_u : U \to \Re .$$

Dentro desse contexto, há várias técnicas de como trabalhar com espaços de busca restritivos, sendo que cada uma delas apresenta uma abordagem distinta. A seguir, será apresentada uma breve discussão dessas técnicas.

4.4.1 Funções com Penalidade

Segundo SMITH & COIT (1997, C5.2), existem dois tipos básicos de funções de penalidade:

- função de penalidade externa (aplicada às soluções infactíveis);
- função de penalidade interna (aplicada às soluções factíveis).

Para o caso de função de penalidade interna, a principal idéia envolvida é o fato do problema apresentar uma solução ótima tal que pelo menos uma restrição seja ativada, o que implica em dizer que a solução ótima está no limiar entre a factibilidade e a infactibilidade. Conhecendo-se essa característica, uma penalidade será aplicada às soluções factíveis quando a restrição não for ativada. Ressalta-se que, para o caso de múltiplas restrições, esse tipo de implementação torna-se bastante complexa.

Para funções de penalidade externa, há três tipos de solução:

- método no qual nenhuma solução infactível será considerada;
- função de penalidade parcial, sendo que uma punição é aplicada nas proximidades do limite de factibilidade;
- função de penalidade global, sendo que uma punição é aplicada na região de infactibilidade.

Dado um problema de otimização, a formulação a seguir caracteriza-se por ser a mais geral:

$$\min_x f(x) \text{ tal que } x \in A \text{ e } x \in B,$$

sendo x um vetor de variáveis de decisão e tal que restrições $x \in A$ são relativamente fáceis de serem atendidas, enquanto restrições $x \in B$ são relativamente difíceis. Como exemplo, o conjunto A pode representar variáveis contínuas e o conjunto B, variáveis discretas.

Assim, o problema pode ser reformulado como:

$$\min_x f(x) + p(d(x,B)) \text{ tal que } x \in A,$$

sendo $d(x,B)$ uma métrica descrevendo a distância do vetor solução x à região B e $p(.)$ sendo uma função de penalidade monotonicamente não-decrescente tal que $p(0) = 0$. Portanto se $x \in B$, então $d(x, B) = 0$, o que leva à ausência de punição.

Conforme SMITH & COIT (1997, C5.2), várias funções para *p(.)* têm sido estudadas, como também diferentes tipos de métricas para *d(.)*, incluindo uma contagem do número de restrições violadas, distância euclidiana entre *x* e *B*, somatório linear das restrições não atendidas ou somatório de funções exponenciais envolvendo as restrições não atendidas.

Encontrar uma função de penalidade que seja eficiente é uma tarefa difícil. Grande parte dessa dificuldade encontra-se no fato de que a solução ótima freqüentemente está no limiar de uma região de factibilidade. Acrescenta-se, ainda, que ao restringir a busca somente às soluções factíveis, ou ao se impor penalidades muito severas, acaba-se dificultando o encontro da solução ótima. Por outro lado, se a punição não for forte o suficiente, tornará a região de busca extremamente ampla e boa parte do tempo do processo será utilizado na exploração de regiões distantes da região de otimalidade.

4.4.2 Decodificadores

MICHALEWICZ (1997, C5.3) afirma que decodificadores oferecem uma opção interessante para as técnicas de computação evolutiva, sendo que um cromossomo (representação do genótipo) fornecerá instruções para um decodificador ou representará a condição inicial para a elaboração de uma solução factível (representação do fenótipo).

Mais formalmente, um decodificador é uma transformação *T* de um espaço de elementos codificados (onde não há infactibilidade) para uma região de factibilidade do espaço de soluções candidatas (onde há regiões factíveis e infactíveis). A Figura 4.2 mostra um exemplo de decodificação, onde a transformação *T* decodifica um ponto *d* em um representante *s* de um subespaço factível.

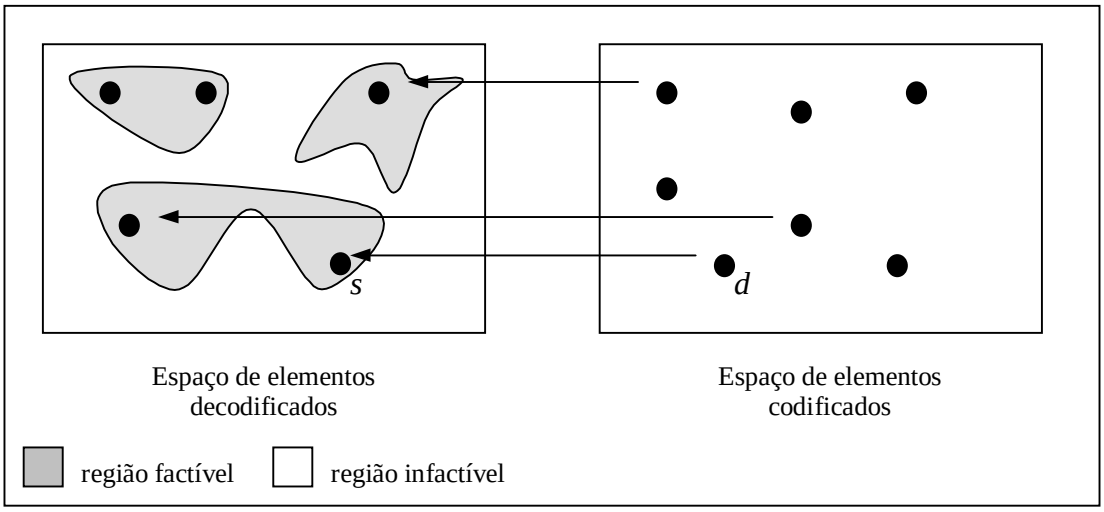

Figura 4.2: Transformação *T* que leva elementos codificados a soluções factíveis.

Será importante considerar que esse processo de decodificação sempre satisfaz as seguintes condições (*F* é o espaço de soluções factíveis):

- para cada elemento $s \in F$, há pelo menos um elemento *d* do espaço de elementos codificados;
- a cada elemento *d* do espaço de elementos codificados corresponde uma única solução factível $s \in F$, ou seja, a transformação *T* é tal que para cada *d* está associado um único $s \in F$.

Também é desejável que a transformação *T*:

- seja computacionalmente rápida;
- deva possuir a característica de que pequenas alterações em elementos do espaço de elementos codificados resultem em pequenas alterações no elemento correspondente do espaço de factibilidade.

4.4.3 Algoritmos de Reparação

Segundo MICHALEWICZ (1997, C5.4), a aplicação de algoritmos de reparação é freqüentemente adotada pela comunidade científica que trabalha com computação evolutiva, acrescentando que para muitos problemas de otimização combinatória é possível fazer com que um indivíduo infactível torne-se factível.

Em relação à função de avaliação, a nova configuração será a seguinte: $eval_u(y) = eval_f(z)$, sendo z a versão factível de y após a aplicação do algoritmo de reparação.

O processo de reparação dos indivíduos pode ser relacionado com a combinação de aprendizado e evolução (efeito *Baldwin* - WHITLEY *et al.*, 1994). O aprendizado é, em geral, uma busca local pela solução factível mais próxima e as modificações ocorridas (o indivíduo infactível é levado para uma região de factibilidade do espaço) serão incorporadas pelo indivíduo ou, então, o indivíduo original não é alterado e um novo elemento será incluído na população com este novo código genético factível.

A desvantagem desse método é a dependência clara em relação ao problema estudado. Em outras palavras, cada problema terá o seu próprio algoritmo de reparação, não existindo um que possa ser generalizado para todos os casos. Acrescenta-se, também, que a heurística utilizada pelo algoritmo deve variar de caso para caso, buscando aquela que melhor se adapta ao contexto.

MICHALEWICZ (1997, C5.4) diz que, para alguns problemas, esse processo de reparação pode ser tão complexo quanto a busca pela solução ótima do próprio problema (em muitos casos, é o que ocorre com problemas de *scheduling*, *timetabling* e outros).

A substituição de indivíduos está relacionada com o lamarckismo (WHITLEY *et al.*, 1994), que assume que um indivíduo passa por um processo de adaptação no decorrer da sua vida no sentido de aumentar a sua adaptabilidade, sendo o resultado deste processo incorporado ao seu código genético. Ainda segundo WHITLEY *et al.* (1994), resultados analíticos e empíricos indicam que estratégias lamarckistas representam um tipo de busca eficaz.

4.4.3.1 Efeito Baldwin

Aprendizado e evolução são considerados processos adaptativos, sendo que o primeiro ocorre durante o ciclo de vida de um indivíduo e é caracterizado pela adaptação a mudanças rápidas no ambiente. Já o segundo processo ocorre durante a história da existência da espécie, sendo caracterizado pela adaptação a mudanças mais lentas no ambiente ou a ambientes estacionários.

Embora seja consenso que não exista interação direta entre aprendizado e evolução, em 1896 James Baldwin propôs um novo fator evolutivo pelo qual características adquiridas poderiam ser herdadas indiretamente. Esse novo fator estaria vinculado ao aprendizado (habilidade que um organismo tem em adaptar-se a seu meio durante a sua vida), sendo denominado de plasticidade fenotípica.

Segundo TURNEY *et al*. (1996), o efeito Baldwin opera em duas etapas distintas:

1. sinergia entre plasticidade fenotípica e evolução: em primeiro lugar, a plasticidade fenotípica permite que um indivíduo se adapte a uma mutação parcialmente bem sucedida (de outro modo seria inútil ao indivíduo). Se essa mutação aumentar a adaptabilidade ao meio, ela tenderá a proliferar-se na população.
2. assimilação genética: ocorrendo a etapa anterior, será dado um certo tempo ao processo evolutivo para encontrar um mecanismo definitivo para a substituição da plasticidade fenotípica.

Como extensões do Efeito Baldwin, percebe-se o seguinte:

- se o aprendizado auxiliar na sobrevivência do organismo, então indivíduos com mais capacidade de aprendizado terão um maior número de descendentes, conduzindo a um aumento da freqüência de genes responsáveis pela habilidade do aprendizado;
- uma vez que o ambiente seja suficientemente estável para que a habilidade mais importante a ser aprendida permaneça a mesma e, ainda, sendo a aprendizagem desta característica custosa (alta pressão seletiva), o processo seletivo poderá indiretamente conduzir à codificação genética desta habilidade (MITCHELL & FORREST, 1995). Em outras palavras, será dado tempo (estabilidade do ambiente) e motivação continuada (custo elevado de aprendizado) para que o processo evolutivo empregue sua criatividade na codificação genética da habilidade desejada. Enquanto essa codificação não ocorre, o indivíduo deverá empregar uma quantidade considerável de recursos para obter esta peculiaridade via aprendizado.
- um comportamento que foi, em um primeiro momento, aprendido pode tornar-se parte de um instinto por intermédio desse processo (desde que seja preservado um nível mínimo de estabilidade no ambiente e que o custo de aprendizagem seja elevado).

De maneira geral, pode-se notar que o efeito Baldwin assemelha-se à teoria lamarckista. Entretanto, no primeiro não há alteração direta do genótipo a partir de modificações do fenótipo do indivíduo. Finalizando, verifica-se que o aprendizado pode afetar a evolução, mesmo quando o que for aprendido não puder ser incorporado diretamente ao genótipo.

4.4.4 Operadores que Preservam as Restrições

Várias pesquisas têm levado à aplicação bem sucedida de operadores genéticos dedicados, tendo a função de preservar a factibilidade dos indivíduos gerados após a sua aplicação (MICHALEWICZ, 1997, C5.5). Percebe-se que essa classe de operadores incorpora um conhecimento específico do problema. Segundo SURRY *et al.* (1995), essa proposta tem a finalidade de construção e utilização de operadores genéticos que tenham a capacidade de atendimento das restrições, de modo que nunca produzam soluções infactíveis.

As principais desvantagens apresentadas por esse processo são:

- operadores genéticos dedicados são aplicáveis somente aos problemas para os quais foram projetados;
- uma análise formal desse tipo de estratégia torna-se difícil, embora uma grande quantidade de experimentos evidenciem a aplicabilidade desta técnica.

4.4.5 Outros Métodos para Trabalhar com Restrições

Várias outras heurísticas para trabalhar com restrições em conjunto com técnicas evolutivas têm sido propostas nos últimos anos. Na maioria das vezes é difícil a classificação desses métodos, uma vez que são a combinação de outros ou são baseados em idéias originais ainda não completamente formalizadas. Dentre essas técnicas podem ser citadas as seguintes (para maiores detalhes veja MICHALEWICZ, 1997, C5.6):

- métodos de otimização multi-objetivo;
- modelos de co-evolução;
- algoritmos meméticos;
- algoritmos genéticos segregados;
- Genocop V (MICHALEWICZ, 2000).

4.5 Abordagens Propostas para o Tratamento de Otimização com Restrições

No desenvolvimento deste trabalho, duas abordagens foram propostas e comparadas no tratamento de problemas de otimização com restrições, particularmente dois problemas de escalonamento. Conforme já apresentado no Capítulo 1, os problemas a serem abordados nos capítulos 5 e 6 caracterizam-se por serem multi-objetivo e por apresentarem múltiplas restrições. De acordo com a nomenclatura adotada neste capítulo, tratam-se de problemas do tipo *COP*: problema de otimização com restrições (*constraint optimization problem*).

A seguir, as duas abordagens utilizadas são apresentadas e posicionadas conforme o contexto deste capítulo (maiores detalhes serão apresentados nos Capítulos 5 e 6).

4.5.1 Codificação Compacta com Algoritmo de Expansão de Código

Na aplicação de algoritmos evolutivos junto a problemas do tipo *COP*, o emprego de uma codificação restrita a um espaço totalmente factível, seguida de decodificação na produção de um elemento factível do espaço original, já foi utilizada na literatura (COSTA, 1999). Essa abordagem será generalizada aqui, com o uso de uma codificação compacta seguida pela aplicação de um algoritmo de expansão de código.

Como é sabido, por um lado, que algoritmos evolutivos em sua composição tradicional encontram dificuldades no tratamento de problemas de otimização com restrições, também é fato, por outro lado, que a aplicação isolada de técnicas de otimização baseada em restrições podem encontrar sérias dificuldades em virtude da existência de ótimos locais de baixa qualidade. Sendo assim, a aplicação conjunta de ambas as metodologias mostra-se promissora pelas seguintes razões:

- o processo evolutivo vai poder atuar somente sobre o código compacto, junto ao qual não há infactibilidade (ou então de modo que a factibilidade possa ser prontamente conquistada), fazendo com que sua eficácia não seja reduzida pela existência de etapas infactíveis;
- embora as técnicas de otimização baseada em restrições, que irão desempenhar o papel de expansão de código, continuem produzindo apenas soluções factíveis com otimalidade local, a existência de uma população de candidatos à solução sendo

evoluída, juntamente com a aplicação de procedimentos de busca local, aumenta significativamente as chances de se obter ótimos locais de boa qualidade.

No entanto, para que esta conjugação de técnicas produza melhores resultados que a aplicação isolada de cada uma delas, é preciso estabelecer critérios para definir o nível de compactação da representação genética, de modo a equilibrar o papel a ser desempenhado por ambas as técnicas. Por exemplo, se a compactação for além do que poderia ser considerado como adequado, o espaço de busca para o processo evolutivo será muito reduzido e o papel da expansão de código passaria a ser muito mais importante. E vice-versa. Existe portanto uma solução de compromisso para o processo de definição do nível de compactação.

Funções de penalidade (seção 4.4.1)

É utilizada uma função de penalidade externa, que não irá permitir nenhuma solução infactível como solução candidata. Este tipo de punição está entre as mais severas, podendo dificultar o encontro da solução ótima caso não venha acompanhada de ferramentas de apoio, como algoritmos de reparação e de busca local.

Decodificadores (seção 4.4.2)

A representação genética compacta irá desempenhar o papel de condição inicial para um algoritmo de expansão de código, que corresponde essencialmente a um decodificador (veja Figura 4.2).

Vale aqui um comentário a respeito da seguinte propriedade que deve estar presente em um decodificador para garantir sua completitude:

- para cada elemento factível $s \in F$ no espaço de elementos decodificados, deve existir pelo menos uma condição inicial no espaço de elementos codificados (representação compacta) de modo que o decodificador seja capaz de produzi-lo.

Como será evidenciado nos capítulos 5 e 6, o algoritmo de expansão de código, que fará o papel do decodificador, não necessita da propriedade de completitude para operar convenientemente, pois existe uma função-objetivo a ser otimizada. Com isso, já durante o processo de expansão de código (e não apenas ao final da decodificação), parte dos

elementos factíveis será descartada por representar um distanciamento da condição de otimalidade. Além disso, ao término da expansão de código, procedimentos de busca local são empregados, de modo que vai haver uma exploração local da região factível, podendo levar a elementos factíveis que melhor atendam aos objetivos, mesmo que estes não sejam diretamente reprodutíveis a partir da decodificação.

O processo de expansão de código pode ser visto como uma busca em uma árvore de decisão, com etapas de *backtracking*. Como a tomada de decisão vai depender da saída de um gerador de números pseudo-aleatórios com a propriedade de repetitividade, então a completitude do decodificador estaria associada à possibilidade de recorrer a todos os possíveis geradores de números psudo-aleatórios, ou seja, a todas as possíveis seqüências pseudo-aleatórias, de modo que qualquer seqüência de ramos da árvore de decisão pudesse ser adotada.

Algoritmos de reparação (seção 4.4.3)

O algoritmo de expansão de código emprega procedimentos de reparação durante o processo de decodificação, e não ao seu final. Portanto, a reparação está incluída no processo de expansão de código. Além disso, procedimentos de busca local são também aplicados imediatamente após o término da etapa de decodificação.

Neste caso, o efeito Baldwin pode representar um papel importante no processo de obtenção da solução, pois é possível optar por condições iniciais ou então geradores de números pseudo-aleatórios que reduzam o custo das etapas de reparação e de busca local.

4.5.2 Codificação Expandida

A busca de solução para problemas *COP* empregando codificação expandida se dá pelos mecanismos usuais de obtenção de indivíduos factíveis via técnicas de otimização baseada em restrições e pela aplicação de operadores evolutivos diretamente sobre o código expandido, seguida da aplicação de procedimentos de reparação e busca local.

São etapas idênticas ao caso anterior, envolvendo expansão de código:

- a técnica de otimização baseada em restrições. No entanto, aqui esta técnica é aplicada uma única vez para cada indivíduo, não necessitando ser reaplicada toda vez que o código genético é alterado, como feito no caso anterior;
- os procedimentos de busca local.

Por outro lado, as diferenças são as seguintes:

- os operadores genéticos aqui só se aplicam ao código expandido, podendo gerar indivíduos infactíveis, enquanto que no caso anterior estavam restritos ao código compacto, gerando sempre indivíduos factíveis. Sendo assim, o genótipo aqui equivale ao fenótipo do caso anterior.
- os procedimentos de reparação aqui só se aplicam sob o código expandido, enquanto no caso anterior eles entram em ação durante a expansão de código.

A expectativa é que o uso de codificação expandida produza 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 baseada 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.