CAPÍTULO 3

GENÉTICO CONSTRUTIVO PARA ROTULAÇÃO

O foco deste trabalho está relacionado à rotulação de pontos usando a proposta do "Constructive Genetic Algorithm" (Furtado, 1998; Lorena e Ribeiro Filho, 1997; Lorena e Furtado, 2001; Ribeiro Filho, 2000; Ribeiro Filho e Lorena, 2001; Oliveira e Lorena, 2002; Lorena, Narciso e Beasley, 2002) para a solução do problema, portanto vamos inicialmente revisar o GA ("Genetic Algorithm") tradicional.

3.1 ALGORITMO GENÉTICO TRADICIONAL

Nesta seção será feito: uma breve descrição histórica de quando e como surgiu o GA, um breve relato sobre as suas características, apresentação do pseudocódigo e descrição dos operadores genéticos de seleção, cruzamento e mutação. As informações descritas nesta seção foram adaptadas dos links que se encontram no final da seção.

3.1.1 HISTÓRICO

Dos tempos de Aristóteles até a metade do século XIX, os naturalistas acreditavam na hipótese da geração espontânea e na criação das espécies de maneira separada por um ser supremo. Mas através de trabalhos como o do naturalista Carolus Linnaeus, que levou a acreditar numa certa relação entre as espécies, influenciaram muitos outros naturalistas a direcionarem-se para a teoria da seleção natural.

Charles Darwin, em 1838, teve a oportunidade de ler o livro do reverendo sociólogo Thomas Robert Malthus intitulado Teoria das populações. Esta obra refere-se a limitações naturais que retringiam o crescimento de uma população devido a fatores ambientais tais como doenças e a fome. Segundo Darwin, foi este livro que despertou nele a inspiração que o levaria à teoria da seleção natural. Após 20 anos de observações e experimentos, em 1858, apresentou a teoria de evolução através da seleção natural, simultaneamente com o naturalista inglês Alfred Russel Wallace.

No início do século XX, os cientistas resgataram o trabalho de Gregor Mendel sobre os princípios básicos da herança genética desenvolvida em 1865, o qual teve enorme influência nos trabalhos futuros relacionados à evolução.

Desenvolvida entre os anos 30 e 40 por biólogos e matemáticos de grandes centros de pesquisa, a teoria moderna da evolução combina a genética com as idéias de Darwin e Wallace sobre a seleção natural, criando um princípio básico de genética populacional: "a variabilidade entre indivíduos numa população de organismos é produzida pela recombinação genética e mutação".

O desenvolvimento dos algoritmos genéticos teve início nos anos 50 e 60 através de muitos biólogos, mas foi John Holland quem começou a desenvolver as primeiras pesquisas no tema. Em 1975, Holland publicou "Adaptation in Natural and Artificial Systems" ponto inicial dos GA's. David E. Goldberg, aluno de Holland, nos anos 80 obteve seu primeiro sucesso em aplicação industrial com GA. Desde então, os GA's são utilizados para solucionar problemas de otimização combinatória e de outras áreas.

3.1.2 TECNOLOGIA GA

GA é um procedimento computacional capaz de emular o processo Darwiniano de seleção natural e mostrou ser bastante interessante em aplicações da otimização combinatória. (Holland. 1975; Goldberg. 1989; Colin. 1995)

Em analogia com o sistema biológico, cada candidato à solução do problema representa o cromossomo ou indivíduo da população e o conjunto de todos os candidatos denomina-se uma geração.

Cada indivíduo da população pode ser representado no GA por um conjunto binário 0 e 1, por números inteiros, por números de ponto flutuante ou por caracteres; e tem um custo associado que determina sua habilidade para sobreviver e produzir descendentes no processo de seleção natural.

O processo de reprodução nos GA's pode ser de simples cópia ou de cruzamento dos cromossomos pais, fazendo com que descendentes de cada geração – ou seja, as novas soluções – sejam semelhantes, possuindo muitas de suas características. Vale ressaltar ainda que os descendentes podem sofrer mutação, ou seja, o resultado dos cromossomos pode ser modificado por perturbações aleatórias.

A forma básica de um GA tradicional é mostrado no algoritmo a seguir:

GA tradicional
t := 0;
Gerar população inicial
Repetir (até alcançar a condição de término)
    Avaliar os indivíduos da população
    Selecionar os melhores indivíduos para reprodução
    Recombinar usando os operadores: cópia, cruzamento ou mutação
    t := t + 1;
Fim_Repetir

3.1.3 OPERADORES GENÉTICOS

A população inicial do GA tradicional é composta por indivíduos criados aleatoriamente ou através dos métodos de inicialização. Cada indivíduo é um cromossomo de tamanho fixo que contém informações codificadas representando normalmente uma solução do problema. Para melhorar o custo ou a aptidão dos indivíduos das populações sucessivas são necessários os operadores genéticos. Esses operadores permitem selecionar indivíduos mais aptos à sobrevivência para reprodução, além de manter as características de adaptação adquiridas pelas gerações passadas, criando indivíduos cada vez mais aptos à sobrevivência. Os operadores genéticos de seleção, cruzamento e mutação estão descritos a seguir.

Indivíduos	Índice de Aptidão	Roleta	Percentagem
I1 10110	22	0 - 22	27%
I2 11001	25	23 - 47	31%
I3 10001	17	48 - 64	21%
I4 00111	7	65 - 71	9%
I5 01010	10	72 - 81	12%

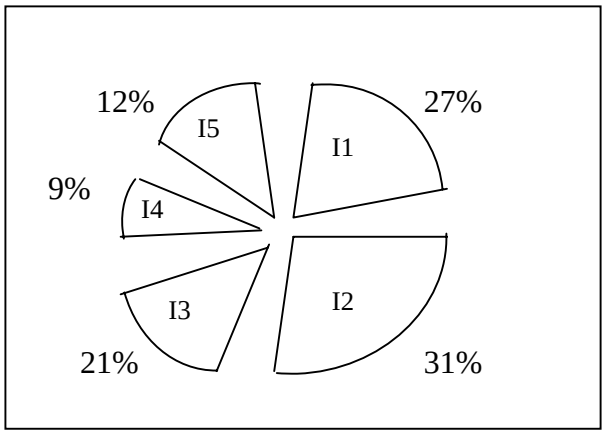

Fig. 3.1 – Esquema da roleta

Seleção: responsável pelo processo de seleção dos indivíduos mais adaptados ao ambiente. O método de seleção muito utilizado é o da roleta, onde cada indivíduo da população é representado na roleta conforme seu valor de aptidão ou custo. Dessa forma, os indivíduos com elevada aptidão receberão um intervalo maior na roleta, enquanto aqueles que têm baixa aptidão receberão menor intervalo na roleta. Após a distribuição, são gerados aleatoriamente números no intervalo entre 0 e o valor da somatória de aptidão de todos os indivíduos da população. O indivíduo que possuir em seu intervalo o número gerado, será selecionado para o cruzamento. A Figura 3.1 mostra o esquema da roleta. A primeira coluna mostra os indivíduos da população, a segunda coluna o valor de aptidão dos indivíduos, a terceira coluna o intervalo de cada indivíduo e a quarta coluna a percentagem correspondente a cada indivíduo. Como exemplo, foram gerados de maneira aleatória, 10 números pertencentes ao intervalo de 0 a 81 (51-1-22-74-46-60-42-65-41-34). Os indivíduos selecionados para o cruzamento foram (I3, I1, I1, I5, I2, I3, I2, I4, I2, I2), levando em consideração que um indivíduo pode ser selecionado mais de uma vez para o cruzamento.

Cruzamento: responsável por geração de novos descendentes. O objetivo do cruzamento é a propagação das características dos indivíduos mais aptos a sobrevivência, por meio de troca de segmentos entre pares de indivíduos, dando assim origem a novos indivíduos. A seguir será descrito algumas formas de reprodução em GA: cruzamento de um ponto, cruzamento de dois pontos e cruzamento uniforme.

Cruzamento de um ponto: o ponto de quebra dos cromossomos pai são escolhidos de forma aleatória, e depois faz-se a troca do conjunto de genes entre os cromossomos para gerar os filhos, como mostrado na Figura 3.2.

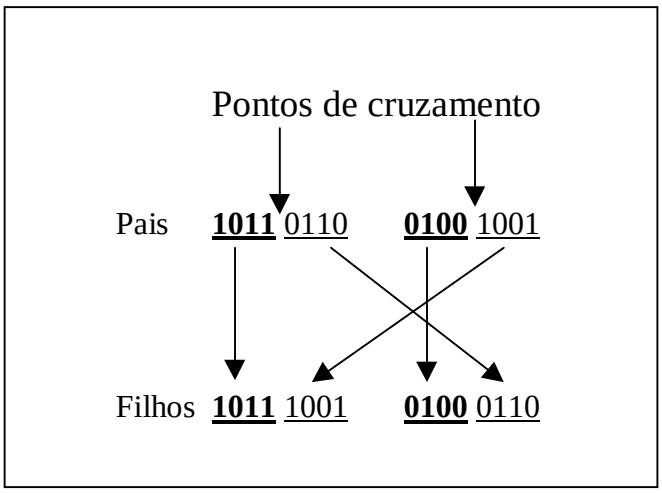

Fig. 3.2 – Cruzamento de um ponto

Cruzamento de dois pontos: similar ao cruzamento de um ponto, mas a troca de segmentos de genes ocorre como mostrado na Figura 3.3.

Cruzamento uniforme: Cada gene do filho é criado, copiando o gene de um dos pais, que é escolhido de acordo com uma máscara de cruzamento gerada aleatoriamente. Se a máscara de cruzamento apresentar 1, o filho recebe gen do primeiro pai, caso contrário recebe gen do segundo pai, como mostrado na Figura 3.4.

Mutação: responsável por manutenção e introdução da diversidade genética na população. Altera aleatoriamente um ou mais genes do cromossomo escolhido, com a

finalidade de introduzir novos elementos na população e melhorar a chance de chegar a qualquer ponto do espaço de busca. A Figura 3.5 ilustra o processo de mutação do filho.

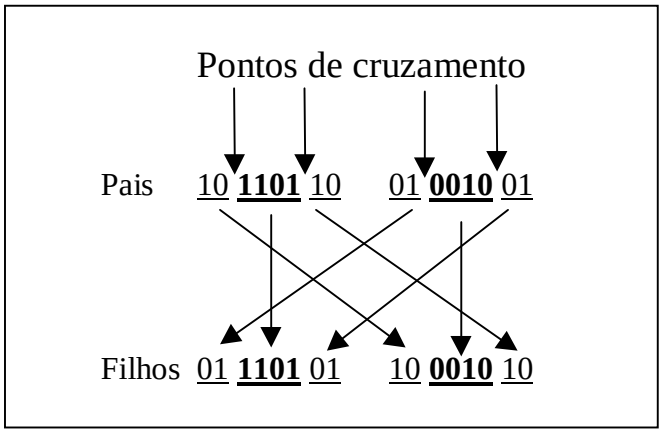

Fig. 3.3 – Cruzamento de dois pontos

Máscara	1 0 0 1 0 1 0 1	1 0 0 1 0 1 0 1
Pai1	1 0 1 0 0 1 0 0	**0 1 1 0 1 0 1 1**
Filhos	1 **1 1** 0 **1** 1 **1** 0	**0** 0 1 **0** 0 **0** 0 **1**
Pai2	**0 1 1 0 1 0 1 1**	1 0 1 0 0 1 0 0

Fig. 3.4 – Cruzamento uniforme

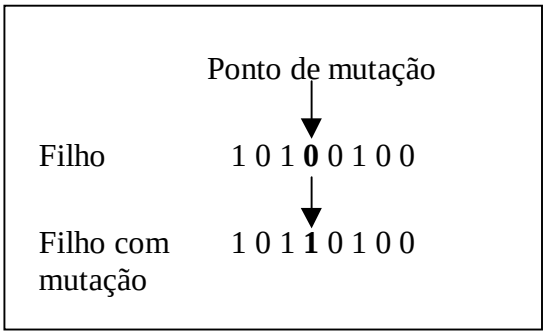

Fig. 3.5 – Mutação

Para informações atualizadas sobre aplicações de GA's veja os links: (http://www.faqs.org/faqs/ai-faq/genetic/), Projeto Isis – Sistemas Inteligentes: Uma Incursão aos Algoritmos Genéticos (http://www.geocities.com/igoryepes/visualizar2.htm), JEICG Home Page – Algorítmos Genéticos (http://www.orbita.starmedia.com/~jeicg/), Algoritmos Genéticos: Fundamentos e Aplicações (http://www.gta.ufrj.br/~marcio/genetic.html), Algoritmo Genético (http://www.sites.uol.com.br/jamwer/GA.HTM) e Mimura e Oliveira, 2002.

3.2 ALGORITMO GENÉTICO CONSTRUTIVO

Atualmente os GA's são muito conhecidos e muito utilizados para solucionar problemas de otimização combinatória por fornecer mecanismos de busca adaptativos, poderosos e robustos, que conduzem a soluções de alta qualidade. Existem muitas variações do GA clássico tradicional, aplicados a diversos problemas combinatórios (Dijk et. al., 1998; Verner et.al.,1997; Lorena e Furtado, 2001; Oliveira A.C.M. e Lorena L.A.N., 2002). Este Capítulo apresenta a aplicação do CGA ("Constructive Genetic Algorithm") ao problema de rotulação de pontos.

O CGA (Furtado, 1998; Lorena e Ribeiro Filho, 1997; Lorena e Furtado, 2001; Ribeiro Filho, 2000; Ribeiro Filho e Lorena, 2001; Oliveira e Lorena, 2002; Lorena, Narciso e Beasley, 2002) difere do GA tradicional descrito na seção 3.1, por avaliar esquemas diretamente e também por apresentar algumas características novas tais como o tamanho dinâmico da população que é composto por esquemas e estruturas, e a possibilidade de usar heurísticas na definição da função objetivo.

Como o objetivo deste trabalho foi a de resolver o problema de rotulação de pontos para gerar mapas de impressão, que devem ser de boa qualidade, foi usado o CGA, que é um algoritmo da família dos GA's e como tal, apesar do tempo de processamento ser grande, costuma trazer resultados robustos de altíssima qualidade.

3.3 TECNOLOGIA CGA

Serão descritos a seguir aspectos de modelagem das representações esquema e estrutura, considerações sobre o problema de rotulação de pontos como um problema de otimização bi_objetivo e o processo de evolução do CGA.

3.3.1 REPRESENTAÇÃO DE ESTRUTURA E ESQUEMA

Em rotulação de pontos, cada esquema representa uma configuração de distribuição de rótulos de um dado conjunto de pontos, onde cada posição do esquema corresponde a um ponto e pode assumir uma das posições candidatas que lhe é permitido ou o símbolo # que informa que o ponto não está temporariamente sendo considerado.

Por exemplo para um problema com 10 pontos a serem rotulados: S = (L1, L4, #, #, L1, L3, L2, L1, #, L2) é um possível esquema, onde L1, L2, L3, L4 refere-se a posições candidatas e o símbolo # implica que o ponto em questão não está sendo considerado temporariamente, assim o esquema S não apresenta soluções candidatas para 3 dos 10 pontos do problema.

As estruturas por representarem soluções completas de um problema, não apresentam o símbolo # na sua composição. Por exemplo: S = (L1, L4, L2, L1, L1, L3, L2, L1, L4, L2) é uma possível estrutura, onde L1, L2, L3, L4 refere-se a posições candidatas.

O CGA tem por objetivo avaliar tanto estruturas, quanto partes dessas estruturas, chamadas esquemas.

3.3.2 FUNÇÕES DE AVALIAÇÃO

O CGA utiliza duas funções, f(S) e g(S) para avaliar a qualidade da rotulação. O f(S) refere-se ao número de rótulos colocados sem sobreposição considerando o esquema S e o g(S) corresponde ao número de rótulos colocados sem sobreposição considerando o esquema S após a aplicação de uma heurística. Para a contagem de conflitos foi utilizado a informação referente ao número de rótulos em conflito (Figura 2.3), pois

alem de mostrar o número total de rótulos que se encontra em conflito, é o tipo de contagem de conflitos que se é adotado na maioria dos artigos encontrados na literatura.

Assim sendo, seja X o conjunto de todas as estruturas e esquemas formadas pelos símbolos {L1, L2, L3, L4, #} para pontos com 4 posições candidatas, temos segundo (Lorena e Furtado, 2001) que, $g : X \rightarrow R_+$, $f : X \rightarrow R_+$, $g(S) \geq f(S)$ e os objetivos são minimizar $\{g(S) - f(S)\}$ e alcançar o estado em que g(S) aproxime o número total de pontos. Assim sendo o CGA é modelado como segue:

$$\begin{aligned} & \text{Min } \{g(S) - f(S)\} \\ & \text{Max } g(S) \qquad (3.1) \\ \text{Sujeito a:} \quad & g(S) \geq f(S),\ S \in X \end{aligned}$$

3.3.3 heurística RSF

A heurística usada para calcular g(S), para o problema de rotulação de pontos, foi denominada RSF ("Recursive Smallest First"). Este algoritmo tenta rotular tantos pontos do mapa quantos forem possíveis, deixando então alguns pontos sem o rótulo para evitar conflitos. O número de pontos sem o rótulo cresce de acordo com o número de pontos sendo mostrado na área do mapa, mas consegue alcançar o estado próximo ao número total de pontos com um tempo mínimo de processamento. A heurística RSF aproxima, portanto a solução ótima do problema de encontrar o maior conjunto independente de vértices no grafo de relacionamentos (ver seção 2.4). A seguir será apresentado o algoritmo RSF adaptado de Strijk et. al. (2000).

Dado uma instância do problema de n pontos, podemos representar o relacionamento das posições candidatas em um grafo G (V,E), onde V = {L1, L2, ..., Ln} representa o conjunto de rótulos, ou seja, cada rótulo está representado por um vértice e E = { A1, A2, ..., Am} representa o conjunto de arestas formados por 2 rótulos que não podem estar ativos simultaneamente por causar conflito ou porque os 2 rótulos pertencem ao mesmo ponto.

Algoritmo RSF

Início:

S := ∅ conjunto independente (solução)

L := V lista de vértices ativos

Enquanto L ≠ ∅

- **Calcular** grau dos vértices de L
- **Escolher** de L, vértice v de menor grau e **acrescentar** em S
- **Tornar** vértice v e vértices adjacentes a v inativos em L

FIM_Enquanto

O algoritmo RSF sempre escolhe do conjunto de vértices ativos, o vértice de menor grau e no caso de haver empate a escolha será pelo vértice (posição candidata) cujo ponto se encontra com menos posições candidatas, uma vez que cada posição candidata pertence a um determinado ponto e o ponto que possui mais posições candidatas tem mais chance de receber um rótulo do que o ponto que possui menos posições candidatas. O algoritmo termina quando a lista de vértices ativos ficar vazia. A seguir um exemplo de funcionamento do algoritmo RSF.

Dados:	ponto	posições candidatas
	P1	L01, L02, L03, L04
	P2	L05, L06, L07, L08
	P3	L09, L10, L11, L12
	P4	L13, L14, L15, L16
	P5	L17, L18, L19, L20
	P6	L21, L22, L23, L24

E a matriz de adjacências descrita na Figura 3.6, e o grau dos vértices.

	L01	L02	L03	L04	L05	L06	L07	L08	L09	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20	L21	L22	L23	L24
L01	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L02	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L03	1	1	0	1	0	1	1	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0
L04	1	1	1	0	1	1	1	1	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0
L05	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
L06	1	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L07	0	0	1	1	1	1	0	1	0	1	1	0	1	1	0	0	1	1	1	1	0	0	0	0
L08	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	1	0	0	1	0	0	0	0
L09	0	0	0	1	1	0	0	1	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0
L10	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	0	0	0	0
L11	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0
L12	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	0
L13	0	0	0	1	0	0	1	1	1	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0
L14	0	0	1	1	0	0	1	0	0	1	1	0	1	0	1	1	1	1	1	1	0	0	0	0
L15	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	1	1	1	1	0	0
L16	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0
L17	0	0	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0
L18	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	0	1	0	1	1	0	0	0	0
L19	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	1	0	1	1	1	0	0
L20	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0	1	0	0	0
L21	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1
L22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	1
L23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
L24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Fig. 3.6 - Matriz de adjacência inicial

Grau dos vértices: L01 = 5, L02 = 4, L03 = 9, L04 = 12, L05 = 8, L06 = 7, L07 = 13, L08 = 11, L09 = 8, L10 = 13, L11 = 14, L12 = 9, L13 = 12, L14 = 12, L15 = 8, L16 = 7, L17 = 14, L18 = 9, L19 = 9, L20 = 12, L21 = 9, L22 = 5, L23 = 3, L24 = 3.

Algoritmo RSF

Iteração 1: L = {L01, L02, L03, ... , L24}

S = {L24} (vértice de menor grau de L)

L23 e L24 são posições candidatas do ponto P6, portanto a escolha entre L23 e L24 é indiferente uma vez que a preferência cartográfica não esta sendo considerada.

	L01	L02	L03	L04	L05	L06	L07	L08	L09	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20
L01	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L02	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L03	1	1	0	1	0	1	1	0	0	1	0	0	0	1	0	0	1	1	0	0
L04	1	1	1	0	1	1	1	1	1	1	0	0	1	1	0	0	1	0	0	0
L05	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0
L06	1	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0
L07	0	0	1	1	1	1	0	1	0	1	1	0	1	1	0	0	1	1	1	1
L08	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	1	0	0	1
L09	0	0	0	1	1	0	0	1	0	1	1	1	1	0	0	0	1	0	0	0
L10	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0
L11	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1
L12	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	1	1	0	0	1
L13	0	0	0	1	0	0	1	1	1	1	1	1	0	1	1	1	1	0	0	1
L14	0	0	1	1	0	0	1	0	0	1	1	0	1	0	1	1	1	1	1	1
L15	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	1	1
L16	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1
L17	0	0	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1
L18	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	0	1	0	1	1
L19	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	1	0	1
L20	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0

Fig. 3.7 - Matriz de adjacência 1ª iteração

Grau dos vértices: L01 = 5, L02 = 4, L03 = 9, L04 = 12, L05 = 8, L06 = 7, L07 = 13, L08 = 11, L09 = 8, L10 = 13, L11 = 13, L12 = 8, L13 = 12, L14 = 12, L15 = 6, L16 = 6, L17 = 14, L18 = 9, L19 = 7, L20 = 11.

Iteração 2: L = {L01, L02, L03, ... , L20}

S = {L24, L02} (vértices de menor grau de L)

	L05	L07	L08	L09	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20
L05	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0
L07	1	0	1	0	1	1	0	1	1	0	0	1	1	1	1
L08	1	1	0	1	1	1	1	1	0	0	0	1	0	0	1
L09	1	0	1	0	1	1	1	1	0	0	0	1	0	0	0
L10	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0
L11	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1
L12	0	0	1	1	1	1	0	1	0	0	1	1	0	0	1
L13	0	1	1	1	1	1	1	0	1	1	1	1	0	0	1
L14	0	1	0	0	1	1	0	1	0	1	1	1	1	1	1
L15	0	0	0	0	0	1	0	1	1	0	1	0	0	1	1
L16	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1
L17	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1
L18	0	1	0	0	1	1	0	0	1	0	0	1	0	1	1
L19	0	1	0	0	0	1	0	0	1	1	0	1	1	0	1
L20	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0

Fig. 3.8 - Matriz de adjacência 2ª iteração

Grau dos vértices: L05 = 5, L07 = 10, L08 = 9, L09 = 7, L10 = 10, L11 = 13, L12 = 8, L13 = 11, L14 = 10, L15 = 6, L16 = 6, L17 = 12, L18 = 7, L19 = 7, L20 = 11.

Iteração 3: L = {L05, L07, L08, ... , L20}

S = {L24, L02, L05} (vértices de menor grau de L)

	L11	**L12**	**L13**	**L14**	**L15**	**L16**	**L18**	**L19**	**L20**
L11	0	1	1	1	1	1	1	1	1
L12	1	0	1	0	0	1	0	0	1
L13	1	1	0	1	1	1	0	0	1
L14	1	0	1	0	1	1	1	1	1
L15	1	0	1	1	0	1	0	1	1
L16	1	1	1	1	1	0	0	0	1
L18	1	0	0	1	0	0	0	1	1
L19	1	0	0	1	1	0	1	0	1
L20	1	1	1	1	1	1	1	1	0

Fig. 3.9 - Matriz de adjacência 3ª iteração

Grau dos vértices: L11 = 8, L12 = 4, L13 = 6, L14 = 7, L15 = 6, L16 = 6, L18 = 4, L19 = 5, L20 = 8.

Iteração 4: L = {L11, L12, L13, L14, L15, L16, L18, L19, L20}

S = {L24, L02, L05, L12} (vértices de menor grau de L)

P3 possui as posições candidatas {L11, L12} e P5 possui as posições candidatas {L18, L19, L20}, portanto L12 foi escolhido ao invés de L18.

	L14	**L15**	**L18**	**L19**
L14	0	1	1	1
L15	1	0	0	1
L18	1	0	0	1
L19	1	1	1	0

Fig. 3.10 - Matriz de adjacência 4ª iteração

Grau dos vértices: L14 = 3, L15 = 2, L18 = 2, L19 = 3.

Iteração 5: L = {L11, L15, L18, L19}

S = {L24, L02, L05, L12, L15} (vértices de menor grau de L)

P4 possui a posição candidata {L15} e P5 possui as posições candidatas {L18, L19}, portanto L15 foi escolhido ao invés de L18.

	L18
L18	0

Fig. 3.11 - Matriz de adjacência 5ª iteração

Iteração 6: L = {L18}

S = {L24, L02, L05, L12, L15, L18}

Iteração 7: L = { }

Na iteração 1 o conjunto L é composto por todos os vértices e o vértice L24 é escolhido de L, por ser o vértice de menor grau. O vértice escolhido é adjacente a vértices L21, L22 e L23 como pode ser visto na matriz de adjacência da Figura 3.6 acima, portanto o conjunto L agora é composto por todos os vértices menos L21, L22 e L23. Na iteração 2, o L2 é o vértice de menor grau, portanto será escolhido e novamente o conjunto L será construído por vértices não adjacentes a L24 e L2. E assim o processamento continua até L ser ∅.

Como pode ser visto no exemplo acima, o conjunto independente máximo foi S = {L24, L02, L05, L12, L15, L18}, ou seja, os 6 pontos foram rotulados sem conflitos: P1/L02, P2/L05, P3/L12, P4/L15, P5/L18 e P6/L24.

Será mostrado a seguir, através de um exemplo de 8 pontos, o calculo das funções f e g para o esquema (L2, L5, L11, L15, L17, L23, #, #). f(L02, L05, L11, L15, L17, L23, #, #) = 3 pontos rotulados sem conflitos, pois L05 conflita com L17, L11 conflita com L15, L11 conflita com L17, L17 conflita com L05 e L17 conflita com L11, como pode ser visto na matriz de adjacência da Figura 3.6. Como pode ser visto no exemplo anterior de 6 pontos, o g(L02, L05, L11, L15, L17, L23, #, #) = {L02, L05, L12, L15, L18, L24} = 6 pontos rotulados sem conflitos. O RSF deste trabalho escolhe o melhor rótulo para cada ponto do esquema

que não apresenta # pois, este símbolo implica que o ponto em questão não está sendo considerado temporariamente. Assim sendo, as posições candidatas L02, L05 e L15 foram mantidas ativas, enquanto que as posições candidatas L11, L17 e L23 foram trocadas por posições candidatas L12, L18 e L24 respectivamente, e o melhor esquema ficou sendo { L02, L05, L12, L15, L18, L24, #, #}, que não apresenta nenhum conflito.

3.3.4 PROCESSO DE EVOLUÇÃO

Outra característica do algoritmo é o tamanho variável da população que é controlado pelo uso de um parâmetro para eliminação de indivíduos ao longo das gerações. Neste contexto, indivíduo refere-se a um esquema ou a cada estrutura candidata a solução do problema e o conjunto de todos os indivíduos denomina-se uma geração. Assim sendo, o indivíduo Si criado recebe um valor de rank $\delta(Si)$ e ele é removido da população quando $\alpha \geq \delta(Si)$. O α refere-se ao tempo de evolução e o valor de rank do indivíduo Si, $\delta(Si)$ é definido como:

$$\delta(Si) = [dg_{max} - [g(Si) - f(Si)]] / d\,[g_{max} - g(Si)] \qquad (3.2)$$

onde gmax é o número total de pontos a serem rotulados.

Quanto maior for o valor de $\delta(Si)$, mais adaptado é considerado o indivíduo Si em relação aos objetivos de minimização $[g(Si) - f(Si)]$ e maximização de $g(Si)$, e mais tempo de sobrevivência para cruzamento ele tem. O parâmetro d é um número real pré-definido $0<d\leq 1$, o número real não negativo g_{max} é o limite superior para $g(Si)$, tal que $g_{max} \geq MAX_{Si \in X}\, g(Si)$ e α está relacionado ao tempo de evolução, começando com o valor zero na primeira geração e sendo acrescido lentamente de $\Delta\alpha$ durante o processo. Assim a população P_α tem seu tamanho influenciado dinamicamente por α e pode eventualmente ficar vazia durante o processo.

Os melhores indivíduos apresentam alto valor de rank, refletindo em pequenas variações de [g(Si) – f(Si)] e/ou grande valor de g(Si), ou seja, os melhores indivíduos são aqueles que são treinados pela heurística RSF e os que apresentam maior número de rótulos sem conflitos.

Analisando o $\delta(Si)$ da equação 3.2, é possível notar o efeito de d e g_{max} no tamanho da população. Quando o valor de d é pequeno, a população apresenta um crescimento lento e se mantém pequena, enquanto que um valor grande de d apresenta uma população grande trazendo problemas de armazenamento da população, mas eventualmente com boas estruturas. O efeito indesejável de g_{max}, é quando $g_{max} >> g(Si)$ para um grande número de indivíduos da população. Isto pode conduzir a eliminação de indivíduos de uma geração para outra e naturalmente, o efeito pode ser uma enorme redução no tamanho da população em apenas algumas gerações. Logo a definição dos parâmetros d e g_{max} precisa ser o resultado de um estudo cuidadoso.

Considerando que os indivíduos bem adaptados precisam ser preservados para o cruzamento e que o tamanho da população P_α no tempo α varia de acordo com o $\Delta\alpha$, o incremento $\Delta\alpha$ tem grande importância no funcionamento do CGA pois, um incremento acelerado demais pode rapidamente eliminar todos os indivíduos da população sem que esses tenham tempo de se reproduzir. Oliveira e Lorena (2002) sugerem variar $\Delta\alpha$ de acordo com o tamanho da população corrente, levando em consideração a iteração em que se encontra e o δ máximo (δ_{top}) e o δ mínimo (δ_{bot}) alcançado. Assim sendo, o incremento adotado neste trabalho foi:

$$\Delta\alpha = k * |P| * ((\delta_{top} - \delta_{bot}) / Gr) + \iota \qquad (3.3)$$

Onde k é uma constante de proporção, ι é o incremento mínimo permitido, Gr é o número restante de gerações e ($\delta_{top} - \delta_{bot}$) é o intervalo corrente dos valores de δ.

3.4 OPERADORES DO CGA

Nesta seção serão descritos os operadores de seleção, recombinação e mutação que trabalham no processo de evolução do CGA. A população inicial é composta exclusivamente de esquemas e a recombinação cria novos esquemas ou estruturas. A melhor estrutura gerada durante o processo de evolução é mantida para representar a melhor solução encontrada e a mutação é usada para diversificar a busca para encontrar melhores soluções.

3.4.1 POPULAÇÃO INICIAL

A população inicial é composta exclusivamente de indivíduos do tipo esquema, e a população das próximas gerações pode aumentar por adição de novos indivíduos gerados através do cruzamento de dois indivíduos.

Cada indivíduo da população inicial é criado com um número de # pré-estabelecido. Os gens que recebem # são escolhidos aleatoriamente. O restante dos gens do indivíduo recebem as posições candidatas geradas aleatoriamente.

A proporção de rótulos do tipo # tem influência direta no processo de evolução. Esquemas que apresentam um número pequeno de #, podem chegar a gerar estruturas durante o cruzamento, mas como os esquemas gerados aleatoriamente não são de boa qualidade, as estruturas encontradas também não o são, com o seu tempo de sobrevivência bastante pequeno. Isto conduz, a diminuição considerável da população em apenas algumas gerações, contribuindo até mesmo para a sua extinção. Por outro lado, se o número de # for grande, a probabilidade de gerar filhos do tipo estrutura é pequena e a probabilidade de serem removidos da população se torna grande, uma vez que a diferença (gmax – g(Si)) da equação 3.2 é maior e novamente as populações futuras podem sofrer enorme redução no seu tamanho em apenas algumas gerações, chegando até mesmo a se extinguir. Logo a escolha do número de # em um esquema, precisa ser feito com cuidado, como descrito na seção 3.6.

A população inicial criada aleatoriamente é composta por esquemas, que na maioria não são de boa qualidade e apresentam f(Si) pequeno. Para garantir uma população razoavelmente grande no início, a diferença [g(Si) – f(Si)] da equação 3.2 deve ser pequena, conduzindo-nos ao uso do algoritmo busca local descrito abaixo para obter o g(Si), pois este tipo de algoritmo conduz a resultados inferiores ao do algoritmo RSF usado para se obter g(Si) dos filhos gerados do cruzamento

O processo de busca local visa trocar, para cada ponto com informação, a posição dos rótulos (por exemplo se o rótulo L4 está ativo no ponto 2 e existe conflito, tenta-se ativar a posição candidata que traz menor número de conflitos) até percorrer o último ponto. Este processo foi repetido 5 vezes para cada esquema, com a finalidade de melhorar um pouco mais o valor de g(si).

Algoritmo busca local

Início:

S := ∅ solução

Repetir (para todos os pontos(pi)/gens do indivíduo)

Se #

colocar # no ponto pi em S

senão

Escolher a posição candidata com menor número de conflitos

Colocar a posição candidata escolhida no ponto pi em S

FIM_ Repetir

Os indivíduos gerados por busca local só serão usados para calcular o valor de g, portanto não fará parte da população em questão. Porém, o melhor esquema será mantido.

Para o exemplo de uma população inicial de 6 pontos, mostrado abaixo, foi estabelecido 2 # para cada indivíduo do tipo esquema gerado. Os 6 pontos e as suas 4 posições candidatas são: P1(L01, L02, L03, L04), P2(L05, L06, L07, L08), P3(L09, L10, L11, L12), P4(L13, L14, L15, L16), P5(L17, L18, L19, L20), P6(L21, L22, L23, L24). A

matriz de adjacência da Figura 3.6 foi usada para verificar os conflitos e o custo refere-se a número de rótulos sem conflitos.

Esquema gerado aleatoriamente	custo	Depois de aplicar a busca local	custo
S1 = (#, L06, L10, L16, #, L22)	2	S1 = (#, L05, L11, L16, #, L22)	4
S2 = (L02, L07, #, L15, #, L21)	2	S2 = (L02, L05, #, L13, #, L21)	4
S3 = (L01, L08, L09, L14, #, #)	2	S3 = (L01, L06, L09, L13, #, #)	2
S4 = (L02, #, L12, #, L20, L24)	2	S4 = (L01, #, L09, #, L18, L21)	4
S5 = (#, #, L12, L16, L19, L22)	4	S5 = (#, #, L09, L13, L18, L21)	4
S6 = (#, L06, L12, #, L20, L23)	2	S6 = (#, L05, L12, #, L18, L21)	4

3.4.2 SELEÇÃO

A cada geração do CGA os indivíduos da população são ordenados em ordem crescente, de acordo com uma chave $\varphi(Si)$, onde:

$$\varphi(Si) = [1 + d(Si)] / \eta(Si) \qquad (3.4)$$

e $$d(Si) = [g(Si) – f(Si)] / g(Si)$$

O parâmetro $\eta(Si)$ é o número de gens com símbolos diferentes de # do esquema Si em questão. A equação 3.4 acima privilegia os esquemas melhor adaptados que possuem $[g(Si) – f(Si)]$ pequeno e $\eta(Si)$ grande, ou seja, esquemas que apresentam menor número de conflitos e próximos de estruturas.

Uma parte do início da população fará parte da sub-população base (S_{base}) e a população guia (S_{guia}) é composta por toda população. O método de seleção para recombinação é do tipo base-guia, onde um indivíduo da sub-população base e um indivíduo da população guia são escolhidos aleatoriamente e então o algoritmo de cruzamento descrito abaixo é aplicado, gerando assim um novo esquema, ou até mesmo uma nova estrutura. O objetivo do S_{guia} é o de conduzir uma modificação dirigida no S_{base}. O

tamanho da sub-população base é um parâmetro do CGA e pode variar de acordo com o tamanho do problema. A Figura 3.12 representa seleção do tipo base-guia.

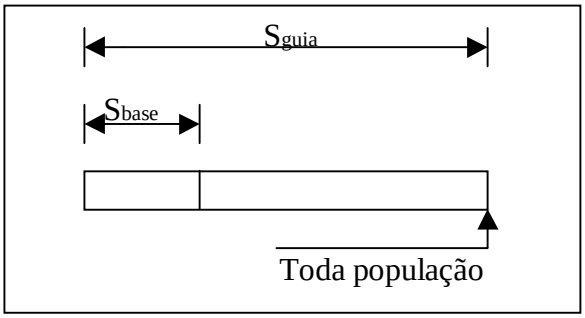

Fig. 3.12 – seleção base-guia

3.4.3 RECOMBINAÇÃO

A recombinação foi feita, usando o algoritmo de cruzamento com máscara, baseado em Verner et al. (1997):

S_{base} um indivíduo da sub-população base

S_{guia} um indivíduo da população guia

S_{novo} um novo indivíduo

M_{base} máscara de S_{base} (0 = conflito, 1 = sem conflito, 2 = #)

M_{guia} máscara de S_{guia} (0 = conflito, 1 = sem conflito, 2 = #)

U 0 ou 1 (gerado aleatoriamente)

Repetir para todos os gens do cromossomo

- Se $M_{base}(j) = 1$ então $S_{novo}(j) \leftarrow S_{base}(j)$
- Se $M_{base}(j) \neq 1$ e $M_{guia}(j) = 1$ então $S_{novo}(j) \leftarrow S_{guia}(j)$
- Se $M_{base}(j) \neq 1$ e $M_{guia}(j) \neq 1$ e $U = 0$ então $S_{novo}(j) \leftarrow S_{guia}(j)$
- Se $M_{base}(j) \neq 1$ e $M_{guia}(j) \neq 1$ e $U = 1$ então $S_{novo}(j) \leftarrow S_{base}(j)$

Após a recombinação, o esquema gerado S_{novo} sofre um processo de busca local descrito na seção 3.4.1 e fará parte da próxima geração se for um sobrevivente, ou seja, se

$\alpha < \delta(S_{novo})$ na equação 3.2 for satisfeito. O número de novos indivíduos sendo criados a cada geração é outro parâmetro do CGA e pode variar de acordo com o problema.

3.4.4 MUTAÇÃO

Após a geração dos filhos, o melhor esquema obtido até o momento e todos esquemas da sub-população base sofrem um processo de mutação. Todos os símbolos # são substituídos por uma melhor posição candidata e depois o algoritmo de busca local descrito na seção 3.4.1 é executado 5 vezes, com o objetivo de melhorar a qualidade destes indivíduos. A melhor estrutura encontrada até o momento é mantida como a melhor solução até então encontrada e se não apresentar conflitos será a solução final. Os mutantes não farão parte da próxima geração. A mutação foi aplicada aos melhores indivíduos da população na tentativa de encontrar uma solução de qualidade.

3.5 ALGORITMO

CGA{Constructive Genetic Algorithm}
Iniciar $\alpha = 0$
Criar P_α
Calcular $f(Si)$, $g^*(Si)$ e $\delta(Si)$ para todo $Si \in P_\alpha$
Repetir (até alcançar a condição de término)
    Se $\alpha < \delta(Si)$, **acrescentar** Si em $P_{\alpha+1}$, para todo $Si \in P_\alpha$
    Estabelecer S_{base} e S_{guia}
    Repetir (número de recombinação)
        Selecionar S_{base} e S_{guia} de P_α
        Recombinar S_{base} e $S_{guia} \rightarrow S_{novo}$
        Aplicar busca local em S_{novo}
        Calcular $f(S_{novo})$, $g(S_{novo})$ e $\delta(S_{novo})$
        Se $\alpha < \delta(S_{novo})$, **acrescentar** S_{novo} em $P_{\alpha+1}$
    Fim_repetir
    Mutação de Sbase e **salvar** a melhor solução

Atualizar α

Fim_repetir

O g*() refere-se ao cálculo do g, usando o algoritmo busca local descrito na seção 3.4.1 e o g() refere-se ao cálculo do g usando o RSF descrito na seção 3.3.3. O melhor g() ou a melhor S_{base} é mantida durante o processo como a melhor solução até então encontrada, como descrito em 3.4.4.

O valor de α é incrementado de $\Delta\alpha$ e o processo é repetido para uma nova geração. O processo pode parar quando atingir um certo número de gerações, quando uma solução satisfatória for obtida, ou quando a população ficar vazia por causa do processo de eliminação de indivíduos.

3.6 CALIBRAÇÃO DE PARÂMETROS

Existem alguns parâmetros que influenciam no funcionamento do CGA e que devem ser especificados a priori. São eles:

- população: Número de indivíduos na população inicial;
- k: constante de proporção;
- ι: incremento mínimo permitido;
- nova solução: quantidade de indivíduos novos criados em uma determinada geração;
- gerações: número de gerações para parada do processo;
- d: um número real $0<d\leq 1$;
- símbolo #: número de símbolos #, nas estruturas que compõe a população inicial.
- sub-população base: tamanho da sub-população base;

Os valores ideais destes parâmetros variam de acordo com o problema sendo tratado, e seu valor é determinado de modo empírico, isto é, o algoritmo é executado algumas vezes até se encontrar valores para os parâmetros que proporcionem resultados

satisfatórios. Para o CGA implementado neste trabalho, os valores dos parâmetros são mostrados na Tabela 3.1 acima.

TABELA 3.1 – SELEÇÃO DE PARÂMETROS

num. de pontos	100 pontos	250 pontos	500 pontos	750 pontos	1000 pontos
população	30	75	150	200	300
k	0.01	0.01	0.01	0.01	0.01
ι	0.00001	0.00001	0.00001	0.00001	0.00001
nova solução	30	30	30	30	30
gerações	25	25	25	25	25
d	0.2	0.2	0.2	0.2	0.2
símbolo #	1	2	10	30	70
sub-população base (% da população)	15%	15%	15%	15%	15%

3.7 RESULTADOS OBTIDOS

Testes foram feitos usando o mesmo conjunto padrão de dados descritos na seção 2.2.3. O algoritmo CGA, implementado em linguagem C++, foi aplicado a 25 configurações diferentes de distribuição de pontos para 100, 250, 500, 750 e 1000 pontos (Apêndice A3). Para cada distribuição de pontos o CGA foi aplicado 6 vezes gerando assim seis amostras para cada configuração. A média obtida da aplicação do CGA nas 25 configurações, onde cada configuração é a média das seis amostras estão mostradas na Tabela 3.2. A média obtida da aplicação do CGA nas 25 configurações, onde cada configuração é a melhor das seis amostras estão mostradas na Tabela 3.3. As linhas das tabelas se referem a:

- Tempo: tempo médio de processamento do algoritmo CGA em segundos para alcançar o estado de não conflito ou de mínimo conflito em um processador Pentium

II 350 Mhz. O tempo mostrado se refere apenas ao tempo de processamento do algoritmo CGA, ou seja, as informações de conflitos são dados de entrada.

- Tempo total: tempo médio de processamento do algoritmo CGA em segundos para processar 25 gerações ou iterações em um processador Pentium II 350 Mhz.

- Com conflito: número médio de rótulos em estado de conflito nas 25 diferentes configurações de distribuição de pontos.

- Desvio padrão: desvio padrão dos rótulos colocados sem conflito.

- Sem conflito: número médio de rótulos em estado de não conflito nas 25 diferentes configurações de distribuição de pontos.

- Sem conflito (%): trata-se da média percentual do número de rótulos sem conflito das 25 diferentes configurações de distribuição de pontos.

TABELA 3.2 – RESULTADOS OBTIDOS POR CGA USANDO O CONJUNTO PADRÃO DE DADOS (MÉDIA)

Num. de pontos	100 pontos	250 pontos	500 pontos	750 pontos	1000 pontos
Tempo	0	0.6	21.5	195.9	981.8
Tempo total	0	0.6	620.8	2178.7	4261.1
Com conflito	0	0	2	24	96
Desvio padrão	0.0	0.0	1.9	6.1	19.8
Sem conflito	100	250	498	726	904
Sem conflito (%)	100.00	100.00	99.6	96.8	90.4

TABELA 3.3 – RESULTADOS OBTIDOS POR CGA USANDO O CONJUNTO PADRÃO DE DADOS (MELHOR)

Num. de pontos	100 pontos	250 pontos	500 pontos	750 pontos	1000 pontos
Tempo	0	0.6	21.5	228.9	1227.2
Tempo total	0	0.6	620.8	2195.8	4243.5
Com conflito	0	0	2	22	93
Desvio padrão	0.0	0.0	1.9	7.1	12.4
Sem conflito	100	250	498	728	907
Sem conflito (%)	100.00	100.00	99.6	97.1	90.7

Os leiautes com configuração de rótulos após aplicação do CGA para 100, 250, 500, 750 e 1000 pontos encontram-se no Apêndice A1.