UM ALGORITMO GENÉTICO BASEADO EM TIPOS ABSTRATOS DE DADOS E SUA ESPECIFICAÇÃO EM Z

Por

Roberta Vilhena Vieira

Tese submetida ao curso de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Augusto César Alves Sampaio
Co-orientador: Manoel Agamemnon Lopes

Recife, 2003

A meus pais Antônio e Glória,
e aos meus irmãos Celina, Luciana e André.

Agradecimentos

A conclusão deste trabalho só foi possível devido à colaboração de muitas pessoas e ao apoio de algumas instituições. Assim, tento, aqui, agradecer a todas elas.

- Ao professor Augusto pela orientação, confiança, apoio, incentivo, e sobretudo pela compreensão, sem os quais eu jamais teria concluído este trabalho.
- Ao professor Agamemnon, da Universidade Federal de Alagoas - UFAL, pela co-orientação, confiança e incentivo; por ter me ensinado a ter sempre uma visão crítica sobre o meu trabalho e pelas discussões que resultaram na proposta do tema desta tese.
- Ao professor Manoel Eusébio pelo incentivo e pela oportunidade de aplicar o algoritmo genético, aqui proposto, a problemas *on line*.
- Ao Centro de Informática da Universidade Federal de Pernambuco pelo ambiente de pesquisa inovador e pelas instalações utilizadas durante o doutorado.
- Aos colegas do Centro de Informática Abel, Aninha, Cristiano, Evandro, Fabíola, Joseane, Luiz, Kelvin, Márcio, Raquel, Renata e Robson pelo companheirismo e convivência. E em particular a Alexandre Cabral também pelo apoio intelectual.
- Aos amigos da UFAL Denise, Eduardo, Eliana e Evandro; e Robério, da Fundação de Amparo a Pesquisa de Alagoas - FAPEAL, pelo apoio e incentivo.
- À Universidade Federal do Pará por permitir o meu afastamento para cursar o doutorado.
- Ao povo brasileiro, que através do CNPq, financiou este trabalho.

Resumo

Este trabalho apresenta um modelo de algoritmo genético baseado em tipos abstratos de dados, denominado de GAADT, no qual o cromossomo é representado por um tipo estratificado em dois níveis de percepção (gene e base), em contra ponto aos demais modelos. A adaptação do cromossomo é comprometida com a relevância das informações codificadas nele. A estratégia de busca do GAADT é altamente objetiva, devido à utilização, como critério de preservação dos cromossomos na população seguinte, de uma função baseada na dinâmica adaptativa da população. A presença explícita do ambiente na funcionalidade do GAADT confere a este algoritmo a capacidade de tratar problemas com alto grau de dinamicidade, como está explorado na aplicação do sistema de monitoramento de sinais vitais de pacientes em unidades de tratamento intensivo de um hospital. Um esboço de uma teoria de processos evolutivos é desenvolvido para descrever a convergência do GAADT, independente da natureza do problema, da representação adotada para o cromossomo, e da população inicial considerada. A aplicação do GAADT a um problema requer a definição dos elementos do ambiente específicos para o problema em foco, os quais devem atender as propriedades estabelecidas na definição do ambiente. A prova de que as definições dos elementos do ambiente, para um dado problema, satisfazem as propriedades exigidas, e que o GAADT quando instanciado para estes elementos satisfaz as propriedades de corretude e aplicabilidade são feitas com o formalismo Z, conferindo assim ao GAADT um rigor matemático. Um estudo comparativo entre a convergência do GAADT com outros modelos é apresentado. As experiências avaliadas neste estudo indicam que o GAADT apresenta maior velocidade de convergência. Por fim, são feitas algumas considerações relevantes sobre o GAADT e sugeridas algumas questões interessantes para trabalhos futuros.

Abstract

This thesis presents a genetic algorithm model based on abstract data types, named GAADT. In this model, the chromosome is represented by a stratified type in two levels of perception (gene and base) which differs it from the other models. The chromosome's adaptation is committed to the importance of the codification it holds. The GAADT's search strategy is highly objective, due the utilization, as a criterion of preservation of chromosomes of the next population, of a function based on the population's adaptability's dynamics. The explicit presence of the environment in the GAADT's functionality, confers to this algorithm the capacity to work with problems in a high dynamic level, as it is explored in the vital signs' monitorship of patients in a Hospital's Intensive Care Units. A sketch of a theory of evolutionary processes is developed to describe GAADT's convergence, despite the problem's nature, the representation adopted to the chromosome and the initial population considered. The application of the GAADT to a problem requires the definition of specific environment elements considering the case in focus, which must observe the established properties. The proof that the environment element's definition to a certain problem, satisfy the demanded/required proprieties and, considering that when the GAADT is instantiated to these problems, it satisfies the properties of certainty and applicability made using the formalism Z, which confers to GAADT a mathematical rigour. A comparative study between the GAADT's convergence and other models is presented.The experiments evaluated in this study indicate that the GAADT presents higher convergence of speed. Finally, some relevant considerations are made on GAADT, and some suggestions for future studies are presented.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Símbolos

t - número de iteração de uma algoritmo genético

p - população do algoritmo genético de Holland e suas variações

h - número de cromossomos selecionados para gerar descendentes no algoritmo genético de Holland e suas variações

$\check{p}$ - população dos cromossomos selecionados para gerar descendentes no algoritmo genético de Holland e suas variações

$\dot{p}$ - população dos cromossomos gerados pela ação dos operadores genéticos de cruzamento e mutação no algoritmo genético de Holland e suas variações

$prob_c$ - probabilidade de cruzamento no algoritmo genético de Holland e suas variações

$prob_m$ - probabilidade de mutação no algoritmo genético de Holland e suas variações

$prob_i$ - probabilidade de inversão no algoritmo genético de Holland e suas variações

m - tamanho do cromossomo no algoritmo genético de Holland e suas variações

$\vec{i}_m^1$ - vetor de dimensão m

w,j,j_1,j_1,... - variável inteira

$\mathcal{G}$ - gramática

$\mathcal{C}_u$ - conjunto dos cromossomos cuja representação adotada é u do algoritmo genético de Holland e suas variações

m - tamanho da população no algoritmo genético de Holland e suas variações

$vida(i)$ - função que define o calculo do parâmetro tempo de vida do cromossomo i do algoritmo genético de Arabas, Mulawka e Pokrasniewicz

$MinLT$ - tempo de vida mínimo permitida no algoritmo genético de Arabas, Mulawka e Pokrasniewicz

$MaxLT$ - tempo de vida máximo permitida no algoritmo genético de Arabas, Mulawka e Pokrasniewicz

η - constante utilizada no calculo do tempo do algoritmo genético de Arabas, Mulawka e Pokrasniewicz

f - função de adaptação do algoritmo genético de Holland e suas variações

$AvgFit$ - valor da adaptação média da população

$MinFit$ - valor da adaptação do cromossomo menos adaptado da população

$MaxFit$ - valor da adaptação do cromossomo mais adaptado da população

$[i]$ - uma lista

$\mathcal{P}_{v_u}$ - conjunto das população com representação v cujos cromossomos são representado pelo tipo u

$\hat{o}$ - função objetiva de um problema

D - domínio de um problema

$decod$ - função que recebe um cromossomo e retorna o elemento do domínio do problema que este cromossomo representa

f_{pen} - função penalidade de um cromossomo

$\mathcal{R}$ - coeficiente de penalidade de um cromossomo

$\lambda(t, i)$ - função heurística do cromossomo i na iteração t

$\vec{\uparrow}$ - função filtro vetorial

$\vec{<}$ - função ordem vetorial

$elitista$ - função de seleção elitista

$rank$ - função de seleção *rank* linear

$power$ - função seleção *rank* não linear

$boltzmann$ - função seleção de *Boltzmann*

$Temperatura$ - grau de entropia da busca realizada pela seleção de *Boltzmann*

$roleta(.)$ - função que define o número de células do vetor contruído para a seleção de Monte Carlo

$roleta(.,.)$ - função seleção de Monte Carlo

$\mathbb{N}$ - conjunto dos números naturais

$\mathbb{Z}$ - conjunto dos números inteiros

$\mathbb{Q}$ - conjunto dos números racionais

$\mathbb{R}$ - conjunto dos números reais

$\mathbb{X}^*$ - conjunto dos números $\mathbb{X}$ sem o zero

$\mathbb{X}^+$ - conjunto dos números $\mathbb{X}$ positivo

$\sharp$ - conteúdo de uma célula de um vetor vazia

$random(w)$ - função de geração de números aleatórios pertencente ao intervalo $[1, w]$

$torneio$ - função seleção de torneio

$maisAdaptado$ - função que recebe um conjunto de cromossomo e retorna um dos cromossomos mais do conjunto fornecida

$seleciona$ - função que recebe uma população e retorna um conjunto de cromossomos

$\vec{+}$ - função soma vetorial

um - função de cruzamento de um ponto de corte

$vários$ - função de cruzamento de vários pontos de corte

$mapa$ - relacionamento entre as informações contidas na mesma posição de dois cromossomos

$comp$ - função que permutação das informações contidas entres duas posições específicas dos cromossomos fornecidos

rem - função que remove as informações repetidas de um cromossomo com a ajuda de um mapeamento que relação as informações contidas no cromossomo fornecido com outro cromossomo

$\mathcal{A}$ - conjunto de símbolos alfanuméricos

$map.2(x)$ - retorna o segundo parânetro da relação map cujo o primeiro parâmetro é x

$parcial$ - função de cruzamento parcialmente ordenado

$\overline{mapa}$ - relação inversa da relação $mapa$

$pres$ - função que completa as informações de um cromossomo com as informações de outro cromossomo preservando a ordem com que elas ocorrêm

$ordem$ - função de cruzamento ordenado

$deriva$ - função de cruzamento por derivação

$\mathcal{G}(z, \vec{z})$ - função que retorna a regra de produção da gramática $\mathcal{G}$ que derivou o símbolo z no vetor $\vec{z}$

and - função de cruzamento por conjunção

or - função de cruzamento por disjunção

a,b - variáveis binárias

$\widehat{+}$ - função lógica $\wedge$

$\widehat{\times}$ - função lógica $\vee$

$complemento$ - função de mutação por complemento de Holland

$inverte$ - função de mutação por inversão de Holland

$troca$ - função de mutação por troca

not - função de mutação por negação

$substitui$ - função de substituição elitista

der - função de substituição deterministica

$vida$ - função de substituição por tempo de vida

s - *schemata*

s_j - elemento do *schemata* s

v_j - variável do *schemata* s

$\vec{s}$ - espaço de busca representado pelo *schemata* s

$const(s)$ - função posição das constantes no *schemata* s

$var(s)$ - função posição das variáveis no *schemata* s

w - número de variáveis de um *schemata*

$\Phi(s, P)$ - função de adaptação do *schemata* s na população p

$card(X)$ - função cardinalidade do conjunto X

$\xi(s, P)$ - função número de indivíduos que pertencem ao *schemata* s na população p

$p_{\triangle}$ - probabilidade de um *schemata* ser preservado pela ação dos operador genético de cruzamento

p_{Ω} - probabilidade de um *schemata* ser preservado pela ação dos operador genético de mutação

p_{∇} - probabilidade de um *schemata* ser preservado pela ação dos operador genético de inversão

Δ - distância entre a primeira e a última ocorrência de uma constante em um *schemata*

$\emptyset$ - conjunto vazio

B - conjunto de base do GAADT

b_j - um elemento da base B

G - conjunto de todos os genes que podem ser formado a partir dos elementos da base B e que satisfazem AFG

g, g_j - elemento do conjunto G

C - conjunto de todos os indivíduos que podem ser formado a partir dos genes do conjunto G e que satisfazem AFC

c, c_j - elemento do conjunto C

P - conjunto de todos os cromossomos

P_j - elemento do conjunto P

Rq - conjunto de requisitos do ambiente

A - ambiente

$grau$ - função grau de adaptação do gene

$adapt$ - função grau de adaptação do indivíduo

$domi$ - função gene dominante

fec - função fecundação

sel_x - predicado seleção

M - caractrísticas do macho

F - característica da femea

$MACHO$ - conjunto dos machos habilitados a cruzarem

$FEMEA$ - conjunto das femeas habiliatdas a cruzarem

$cruz$ - função cruzamento

ins - função inserção

del - função supressão

$troc$ - função troca

mut - predicado mutação

$\mathbb{P}(P)$ - conjunto potência de P

Tx - relação taxonômica

Σ - conjunto de operadores genealógicos que atuam sobre a população P_j

P_0 - população inicial

$\equiv_C$ - equivalência cromossomica

$\equiv_E$ - equivalência evolutiva

$\subseteq_E$ - relação de inclusão evolutiva

PE - processo evolutivo

$\cap_E$ - relação de união evolutiva

$\cup_E$ - relação de interseção evolutiva

$-_E$ - relação de diferença evolutiva

Abreviações:

TAD - tipos abstratos de dados

$GAADT$ - *genetic algorithm by abstract data types*

$FPGA$ - *field programmable gate array*

AFG - axiomas de formação de genes

AFC - axiomas de formação de cromossomos

UTI - unidade de tratamento intensivo

Capítulo 1

Introdução

1.1 Motivação Histórica

Na década de setenta, um estudante de doutorado em ciência da computação da Universidade de Michigan, chamado John H. Holland, tentava desenvolver um método computacional que se prestasse para abordar fenômenos gerados por sistemas adaptativos complexos. Os fenômenos gerados por sistemas adaptativos complexos são aqueles cujos resultados dependem das interações não lineares entre vários agentes adaptativos. Por exemplo, a seca do nordeste brasileiro é um fenômeno desse tipo.

No decorrer do seu trabalho, Holland percebeu que existia uma nítida semelhança entre os fenômenos que estudava e o processo de evolução das espécies, pois assim como a interação entre os agentes adaptativos determinava o resultado dos fenômenos investigados por ele, a interação entre os fatores ambientais determinava a próxima população de uma dada região. Com base nesta constatação, Holland propôs um método computacional para simular o processo de evolução das espécies, denominado de algoritmo Genético [79].

Na percepção de Holland (Capítulo 2), o processo de evolução biológica se resume na transformação de uma população em outra, pela ação dos operadores de seleção, cruzamento, mutação, inversão e substituição. Nesta percepção, a população é um vetor de tamanho fixo de cromossomos, que também são vetores de tamanho fixo de elementos pertencentes ao conjunto {0,1}. Holland denomina de *locu* as posições do cromossomo, *alelo* o valor que ocupa um *locu* no cromossomo e *gene* ao conjunto de *alelos* que podem ocupar um *locu* no cromossomo.

Os operadores do algoritmo genético de Holland se comportam da seguinte forma. Primeiro, a operação de seleção escolhe os cromossomos da população que podem gerar descendentes para a próxima população. A seguir, as operações de cruzamento, mutação e inversão são aplicadas aos cromossomos selecionados com

o objetivo de gerar novos cromossomos. Por fim, a substituição constrói a nova população com alguns dos cromossomos da população atual e da população de cromossomos gerados.

Uma primeira tentativa de descrever formalmente o comportamento do algoritmo genético de Holland foi a teoria dos *schemata* (Capítulo 5) desenvolvida por Holland [79]. Segundo a qual as homologias existentes entre os cromossomos mais adaptados da população têm uma alta probabilidade de estarem presentes na população subseqüente. Estendendo esta idéia a todas as populações construídas durante a execução do algoritmo genético, concluiu-se que as homologias nos cromossomos mais adaptados em cada uma destas populações contribuem para a construção dos cromossomos que compõem a população final.

Mais recentemente, o nome algoritmo genético tem sido usado para nomear um campo de pesquisa da computação evolutiva [58] que compreende tanto os algoritmos inspirados no algoritmo genético de Holland como os estudos teóricos desenvolvidos com o propósito de explicar o comportamento e a convergência destes algoritmos. A computação evolutiva é um ramo de pesquisa da inteligência artificial que surgiu no fim dos anos cinqüenta e início dos anos sessenta. Baseado na proposta de usar a idéia de evolução existente na teoria de Darwin, é um ambiente que se presta ao desenvolvimento natural de métodos computacionais capazes de solucionar problemas de otimização, funcional e combinatorial, complexos e dinâmicos, uma vez que os métodos tradicionais de otimização, com seus modelos matemáticos rígidos, não conseguiam tratar estes problemas [7]. Os campos de pesquisa da computação evolutiva compreendem, além do algoritmo genético, já citado, a programação evolutiva concebida por Fogel [106] como uma estratégia de otimização estocástica que enfatiza o relacionamento comportamental entre os cromossomos progenitores e sua descendência.; a estratégia evolutiva introduzida por Rechernberg [131] como um método capaz de otimizar parâmetros com valores reais; o sistema classificador proposto por Holland [80] como um sistema capaz de perceber e classificar os acontecimentos de um ambiente e reagir a eles de maneira apropriada e, a programação genética apresentada por Koza [103] como um sistema que tem competência de encontrar uma função para descrever um dado fenômeno a partir de um conjunto de fatos conhecidos e de operações válidas para o contexto do fenômeno referido.

Quanto aos algoritmos derivados do algoritmo de Holland, eles baseavam-se na alteração da representação do cromossomo e da população, bem como do comportamento da operação de seleção, cruzamento, mutação e substituição, já que a operação de inversão deixou de ser utilizada por ser considerado, um caso particular da operação de mutação. Alguns dos Algoritmos derivados são brevemente apresentados a seguir.

Smith [151] declara que uma representação para o cromossomo por um vetor binário de tamanho fixo

impede a aplicação do algoritmo genético a um problema[1] no qual estes elementos devem modelar objetos matemáticos complexos. Esta mesma conclusão foi obtida por De Jong [91] ao tentar utilizar o algoritmo genético de Holland a um problema onde o cromossomo terá que modelar uma estrutura hierárquica cuja representação gráfica é desconhecida. Goldberg [62] chegou a resultados semelhantes ao analisar problemas em que os valores dos alelos de alguns locus no cromossomo eram dependentes entre si. Koza [104] também obteve os mesmos resultados ao investigar problemas de aprendizagem de funções booleanas. As propostas de representação do cromossomo motivadas por estes estudos baseavam-se tanto na liberação do gene [4], como na variação do tamanho do cromossomo [5]. Para manipular os dados codificados em cada uma destas representações foram sugeridos novos operadores de cruzamento e mutação [152, 92, 48, 145].

O reflexo das alterações propostas para a representação do cromossomo ensejou a extensão ou elaboração de fundamentos teóricos adequados para explicar o comportamento do algoritmo genético frente a esta nova realidade. Um dos primeiros trabalhos nesta área foi apresentado por Goldberg [57], que argumentava ser a representação binária mais apropriada para o trabalho realizado pelo algoritmo genético do que a representação não binária sob a lente da teoria de *schemata*. Antonisse [4] amplia a interpretação de *schemata* e argumenta que a representação não binária é mais adequada para o algoritmo genético do que a binária. Tade e Smith [158] definem o conceito de convergência esperada do *alelo*, segundo a qual os *alelos* pertencentes a um *gene* com uma cardinalidade grande aparecem distribuídos aleatoriamente dentro da população. Similar argumento é apresentado por Reeves [132], ao dizer que para um *gene* com grande cardinalidade o tamanho da população tem que ser muito grande para a convergência ocorrer. Um possível ponto final no debate sobre qual o melhor conjunto de *genes* para o algoritmo genético é a teoria Mutary de Field [50] que demonstra não existir uma cardinalidade para o *gene* capaz de produzir uma representação do cromossomo ótima, mas sim indicações de como escolher uma cardinalidade para o *gene* que torne a sua representação por um vetor de tamanho fixo ótima quando aplicada a um dado problema.

Outro problema relacionado com a representação proposta por Holland é a influência que o tamanho da população tem sobre a qualidade dos cromossomos da população retornada pelo algoritmo genético [76, 113, 121, 101, 75]. O primeiro registro deste problema foi reportado por Grefenstette [68] ao usar um algoritmo genético para definir o valor ideal dos parâmetros do algoritmo de Holland na abordagem de um problema,

[1]Problema - é uma relação entre dados e resultados adequados aos mesmos segundo uma intenção. Neste sentido uma solução é uma forma de busca de resultados correspondentes a dados que satisfaça a intenção do problema [111, 2]

o qual foi reforçado pelos estudos de Jog *et al.* [86] ao analisar os efeitos do tamanho da população para problemas de otimização combinatorial, e de Cartwright [28] ao investigar os efeitos do tamanho da população sobre problemas de diferentes naturezas por olhar para o algoritmo genético como um algoritmo de espaço de dados. Para contornar a perturbação causada por uma escolha inapropriada do tamanho da população, novas representações de tamanhos variados para a população têm surgido, como no *Messy* de Smith [150] e no *GAVaPS* de Arabas *et al.* [6, 119]. A justificativa teórica para estas abordagens encontra-se nos trabalhos de Goldberg, que considera o número de *schemata* presentes na população necessário para garantir tanto a construção de um cromossomo com uma dada qualidade de adaptação [56], como que o tempo requerido para o algoritmo encontrar este cromossomo seja praticável [58].

Os mecanismos de seleção dividem-se em duas categorias. Em uma categoria o número de cópias de um cromossomo na população de cromossomos escolhidos para gerarem descendentes é igual ao valor da adaptação do cromossomo em uma determinada escala (seleção explícita). Exemplos deste tipo de seleção são a seleção de escalar [8], a seleção de *ranking* [8], a seleção de estado fixo [163, 156] e a seleção de Monte Carlo [58, 89]. Na outra categoria, o número de cópias de um cromossomo é obtido através do uso de métodos probabilísticos e aleatórios (seleção implícita), como na seleção torneio [21]. O respaldo teórico destes mecanismos está provado por Baker [17] para a seleção de ranking; e por Goldberg para a seleção de *ranking* linear em [58], e para vários mecanismos de seleção implícito e explícito em [65].

Mecanismos similares à seleção são propostos para a substituição, tais como substituição de *gap* em que a população inteira é substituída pelos cromossomos mais adaptados gerados pela ação dos operadores de cruzamento, mutação e inversão [68]; de estado fixo, onde somente uma pequena porção (tipicamente dois cromossomos) da população é substituída [38]; de determinismo, onde os cromossomos gerados só substituem os pais se estes apresentarem adaptação menor do que a dos seus filhos [112]; etc.

Apesar de todo o esforço em contornar os problemas apontados no trabalho de Holland, ainda existem muitos destes problemas que permanecem até o dia de hoje sem uma resposta, como o problema do tempo de convergência, da convergência prematura e da epistasis. O problema do tempo de convergência está relacionado ao tempo gastos por este algoritmo para produzir resultados de melhor qualidade do que outros métodos computacionais. As soluções apresentadas para este problema objetivam explorar, ao máximo, as possibilidades de paralelizar e distribuir a execução do algoritmo, como nos algoritmos genéticos paralelos [77, 12, 114, 23, 33], nos algoritmos genéticos de tempo contínuo [161, 7, 154] e nos *hardware* evolutivos [67, 134, 147]. Quando

a complexidade está associada apenas ao cálculo da adaptação do cromossomo, uma versão simplificada da mesma pode ser a solução, como nos algoritmos genéticos baseados em técnicas de relaxamento [140, 1].

A convergência prematura acontece quando as informações contidas nos cromossomos mais adaptados da população não são ótimas para o problema, os quais passam a predominar nas próximas populações geradas pelo algoritmo genético, direcionando assim este algoritmo para uma população de cromossomos ótimos locais. Uma forma de evitar a convergência prematura é manter a diversidade da população, que pode ser conseguida através de mecanismos de controle da diversidade [148, 149], de *sharing* e *niching* [11, 73, 120, 82, 64], e de regeneração da diversidade [47, 49, 53, 115, 160].

A epistasis ocorre quando a adaptação de um cromossomo depende fortemente da interação entre as informações presentes em sua representação [93]. A dificuldade em tratar problemas com epistasis pode ser de natureza representacional e teórica. Dependendo do problema, o custo para desenvolver um novo projeto de representação para o cromossomo livre de epistasis pode ser muito grande [162], assim uma alternativa para tratar este problema é adotar uma nova forma de representação para o cromossomo e desenvolver operadores genéticos de cruzamento e/ou mutação adequados [35]. Os melhores resultados para estes problemas são os algoritmos genéticos desenvolvidos para resolver uma aplicação específica, os quais atendem pela denominação de algoritmos genéticos competentes [123, 96, 62, 74, 100].

A dificuldade teórica da epistasis é que, pela hipótese do bloco de construção[2], a convergência do algoritmo genético só pode ser garantida se o grau de epistasis do problema for baixo [58]. O primeiro passo no sentido de superar esta dificuldade foi a definição formal dos problemas com grau de epistasis baixo e alto apresentada por Davis e Coombs [39]. Para problemas de otimização combinatorial, a dificuldade teórica da espistasis foi totalmente eliminada por Goldberg [63], ao introduzir o conceito de *o-schema* na teoria dos *schemata* de Holland.

1.2 Motivação da Tese

A idéia de se utilizar a lógica evolutiva sugerida pela Teoria da Origem das Espécies de Darwin para resolver problemas complexos, além de epistemologicamente interessante, é muito promissora no sentido de se

[2]Bloco de construção - é um *schema* de pequeno comprimento, consistindo de um conjunto de símbolos do *gene* utilizado para representar o cromossomo, que apresenta uma melhor adaptação quando juntos do que sozinhos. Por este motivo o bloco de construção tende a gerar cromossomos com alta adaptação.

prestar para abordar uma grande quantidade de problemas que oferecem um elevado grau de dificuldade. O desenvolvimento de algoritmos genéticos baseados nesta idéia tem se tornado, ao longo do tempo, um mecanismo computacional adequado ao tratamento de problemas dentro de ambientes complexos evolutivos, com destaque para os ambientes que oferecem uma grande biodiversidade. O nosso planeta é, sem dúvida, uma grande fonte de problemas desta natureza: problemas climáticos e pluviométricos, problemas de controle de tráfego de trânsito, nas grandes cidades, problemas de aproximação de funções, problemas de distribuição de grandes empresas de abastecimento, problemas de monitoramento de pacientes em UTI's hospitalares, entre muitos outros.

Diante de algumas questões importantes da construção dos algoritmos genéticos, como a "limitação" do tamanho fixo do cromossomo e também do tamanho fixo da população, bem como a artificialidade da operação de substituição para renovação das populações geradas, como em Holland e seus seguidores mais imediatos [79, 59, 119], surgiu a motivação para desenvolver um novo modelo de algoritmo genético baseado em Tipos Abstratos de Dados (TAD). Os aspectos que se pretende contribuir com o algoritmo genético baseado em tipos abstratos de dados (GAADT) são principalmente a liberação desses algoritmos da restrição sobre a representação fixa do alfabeto 0,1 e do tamanho fixo da população, bem como a redução das operações genéticas a somente duas: cruzamento e mutação.

A adoção da abordagem de TAD, para construção de algoritmos genéticos permite, entre outras coisas, o desenvolvimento de uma sistemática de especificação do mesmo a partir do estudo dos componentes básicos do problema, para o qual o algoritmo genético a ser desenvolvido é solução. Espera-se que isto represente um avanço na pragmática de uso dos algoritmos genéticos e na amplitude do espectro de aplicação dos mesmos. Dentro da visão de que o algoritmo genético é solução de um problema, tal perspectiva é contraproducente com respeito à idéia de busca dos melhores (mais adaptados) cromossomos (resultados).

A questão da representação binária do cromossomo apresentada por Holland parece levar críticas sobre a pré-definição da estrutura de dados a ser adotada, no caso vetor de tamanho fixo, que está no sentido inverso do percurso natural adotado pelos engenheiros da computação na busca de uma solução computacional para um problema, onde primeiro se estuda o problema para só depois definir a estrutura de dados utilizada para modelar os objetos do domínio do problema. Neste sentido, a representação adotada pelo GAADT permite que um estudo inicial do problema seja feito, e em função deste, e só depois, já em um momento operacional, uma representação adequada seja adotada. Outro importante ganho potencial com a visão acima referida é a

portabilidade das soluções via algoritmo genético, uma vez que o mesmo “esquema de solução” pode resolver uma família de problemas, dependendo da natureza e do ambiente dos mesmos.

1.3 Objetivos e Contribuições da Tese

Este trabalho apresenta o GAADT como uma resposta para os problemas citados na seção anterior, inspirado na interpretação da autora, na visão de um biólogo naturalista [34, 66] e na visão de um geneticista [52, 10], sobre o processo de evolução das espécies, descrita através da linguagem matemática da teoria dos conjuntos. A escolha desta linguagem deve-se ao fato dela ser uma linguagem bastante difundida dentro da comunidade científica para a qual é dirigido o presente trabalho.

A abordagem para a representação do cromossomo proposta é baseada em dois tipos abstratos de dados, o gene e a base, os quais conferem ao cromossomo a flexibilidade de modelar diferentes estruturas de dados dependendo do modo como estes tipos forem especificados para o problema. Na metáfora adotada, a base é o alfabeto, com o qual se escrevem as sílabas (genes) da palavra (cromossomo).

A representação estratificada do cromossomo também permite que se olhe para dentro do mesmo e possa-se predicar sobre a relevância de um dado gene para o cromossomo, percepção esta explorada pela operação de cruzamento apresentada, que trás como novidade uma definição dos pares de cromossomos habilitados ao cruzamento, uma relação dos genes que expressam o mesmo atributo e uma qualificação do gene que expressa um dado atributo com maior relevância para o problema.

O nascimento e a morte dos cromossomos obedecem ao critério de ponto de corte, que nada mais é do que uma medida baseada no valor da adaptação do cromossomo. O nascimento de um cromossomo ocorre sem nenhuma restrição, ou seja, todos os cromossomos gerados, por cruzamento ou mutação, irão fazer parte da próxima população. Os cromossomos destinados a morrer são submetidos ao operador de mutação e registrados na população de cromossomos mortos. A aplicação da mutação sobre os cromossomos destinados a morrer é uma forma de permitir que os genes relevantes para o problema destes cromossomos permaneçam na população através de seus descendentes, enquanto o registro na população de cromossomos mortos é usado como uma memória biológica para evitar que cromossomos já avaliados e descartados possam reaparecer na população, o que resultaria na diminuição da qualidade do resultado encontrado pelo algoritmo genético. Entretanto, para viabilizar todas estas novidades é necessário que a representação da população seja uma estrutura com tamanho

finito em cada momento, mas não limitada, e que o número de ocorrências de um cromossomo na população não afete o comportamento do algoritmo. No presente trabalho, a estrutura adotada foi conjunto.

A presença do formalismo matemático e do formalismo em Z, neste trabalho, se deve ao fato de que a evolução, das idéias que deram origem ao mesmo, foi desenvolvida dentro de uma visão algébrica e só posteriormente, para se verificar a consistência computacional das especificações originadas, foi adotado o formalismo em Z, mais apropriado para esse intuito.

Uma especificação formal em Z [84] do GAADT serve também para ajudar o usuário a projetar aplicações em consonância com as propriedades do algoritmo proposto. Usando uma notação formal como Z é possível caracterizar o GAADT como um arcabouço genérico com relação à representação. A solução de um problema específico deve ser, obrigatoriamente, uma instância (e, ao mesmo tempo, um refinamento) do arcabouço. Exemplos foram desenvolvidos como forma de validar o arabouço proposto. Esta estratégia permite que cada decisão do processo de concepção da aplicação em desenvolvimento seja provada mecanicamente, com a utilização da ferramenta Z_Eves [117, 116]. A escolha da linguagem Z fundamenta-se no fato desta linguagem alicerçar-se na teoria dos conjuntos, utilizada para descrever os tipos abstratos de dados manipulados pelo algoritmo, e na lógica de primeira ordem [126, 153].Um apanhado mais detalhado sobre o formalismo Z pode ser encontrado no Apêndice A ou nas referencias [117, 116].

A teoria de processos evolutivos tem sua origem na constatação de que a teoria dos schemata e suas extensões não podem ser usadas para provar a convergência do GAADT, uma vez que estas teorias são comprometidas com a estrutura de dados utilizada para representar a população e o cromossomo. Asssim, foi desenvolvida uma teoria de processos evolutivos na qual a única exigência para o uso desta teoria é que o algoritmo manipule uma população de qualquer representação de cromossomos modelados por qualquer estrutura, para os quais existe uma função de adaptação. O fundamento computacional da teoria proposta encontra suporte no conceito de monotonicidade e no teorema do ponto fixo de Task para funções monotônicas, ou seja, a monotonicidade estabelece a direção constante de uma busca ganhadora através do processo em pauta e a existência de ponto fixo, significa que a busca de um objetivo satisfatório tem sentido, porque tal grau de satisfação será alcançado (eficácia).

Resumindo, os objetivos deste trabalho são:

- propor um modelo de algoritmo genético baseado na construção de Tipos Abstrato de Dados, denominado de GAADT;

- algoritmo genético GAADT resultante da interpretação dos tipos é sensível ao contexto onde está inserido o problema, ou seja, sensível ao ambiente;
- dado um problema $Prob$ é possível estabelecer uma instância interpretada do modelo GAADT$Prob$ para este problema; e
- o algoritmo genético GAADT instanciado para um problema é convergente independentemente da população inicial.

Não fazem parte, entretanto, dos objetivos deste trabalho: o estudo da complexidade, do ponto de vista do esforço computacional, do GAADT; a análise do grau de eficiência do mesmo, um estudo das características dos problemas solúveis através do GAADT, nem uma taxonomia de tais problemas.

1.4 Organização da Tese

Para cumprir os objetivos acima, organizou-se este trabalho em sete capítulos, incluindo esta introdução, além de quatro Apêndices.

No Capítulo 2, serão descritos modelos de algoritmos genéticos de propósito geral, modelos de algoritmos genéticos desenvolvidos para trabalhar sobre problemas considerados difíceis pelos algoritmos genéticos de propósito gerais e modelos de algoritmo genéticos comprometidos em diminuir o tempo de convergência.

No Capítulo 3, será apresentado o algoritmo genético proposto neste trabalho, baseado em tipos abstratos de dados, e sua especificação em Z. Inicialmente, serão mostrados os tipos abstratos manipulados por este algoritmo. A seguir, serão definidas as operações genéticas utilizadas para promover a adaptação da população pelo algoritmo proposto. Por fim, será descrito o ambiente ao qual os cromossomos do GAADT devem-se adaptar.

No Capítulo 4, será apresentada a teoria de processos evolutivos e a prova segundo esta teoria de que o GAADT é um processo monotônico com ponto fixo, ou seja, que ele converge.

No Capítulo 5, será apresentada uma metodologia para a aplicação do GAADT e alguns exemplos de como esta metodologia pode ser usada para garantir uma instanciação correta do algoritmo a diferentes problemas.

No Capítulo 6, serão mostrados resultados obtidos por implementações do GAADT, de acordo com as especificações apresentadas nos capítulos anteriores, e uma comparação da convergência deste algoritmo com

alguns dos algoritmos descritos no Capítulo 2 aplicados aos mesmos problemas.

No Capítulo 7, serão apresentadas a conclusão desta tese e as perspectivas de trabalhos futuros.

No Apêndice A, será introduzido as noções básicas para entender a especificação na linguagem Z apresentadas nos Capítulos 3 e 5.

No Apêndice B, será detalhado o funcionamento do dispositivo FPGA, utilizado na definição de *hardware* evolutivo.

No Apêndice C, serão exibidas as provas das propriedades exigidas para os tipos, as funções, as relações e as operações utilizadas na definição do GAADT.

No Apêndice D, serão mostradas as provas das propriedades exigidas para os tipos, as funções, as relações, as operações e do refinamento (inicialização, aplicabilidade e corretude) das instanciações do GAADT apresentadas no Capítulo 5.

Capítulo 2

Uma Visão Geral de Algoritmos Genéticos

2.1 Introdução

Algoritmos genéticos trabalham sobre uma população de possíveis resultados (cromossomos) para um problema, os quais são melhorados através de várias iterações com o intuito de gerar um cromossomo ótimo para o problema. O mecanismo de transformação de uma população em outra obedece ao princípio de seleção natural, descrito por Darwin [34], segundo o qual, na natureza, os indivíduos de uma população competem entre si por recursos tais como comida, água e abrigo. Os indivíduos de uma espécie que apresentam as características mais favoráveis ao meio ambiente, denominados de indivíduos adaptados, têm maior chance de sobreviver que os indivíduos que não apresentam estas características. Os indivíduos adaptados são preferidos pelos indivíduos da sua espécie para procriar, gerando assim um maior número de descendentes potencialmente adaptados ao meio ambiente, uma vez que as características apresentadas por um indivíduo são o resultado da combinação das características herdadas dos seus pais. A Figura 2.1 apresenta um pseudo código para os algoritmos genéticos.

Segundo este pseudocódigo, um algoritmo genético tem início com a geração da população inicial p composta por n cromossomos, que podem ser fornecidos pelo usuário, gerados aleatoriamente ou construídos por um algoritmo baseado no conhecimento existente sobre o problema a ser solucionado. Em seguida, é calculada a adaptação dos cromossomos da população, para verificar a qualidade destes cromossomos para o problema investigado. Se o critério de parada definido para o problema não for satisfeito, então um novo ciclo incrementa o contador de iterações t, seleciona cromossomos para gerar descendentes, gera descendentes, nascimento de alguns dos cromossomos gerados e morte dos cromossomos existentes na população considerados de baixa qualidade para o problema analisado tem início. Os critérios de parada usados pelos algoritmos genéticos são encontrar um cromossomo com adaptação *x* ou atingir a *j-ésima* iteração.

$t \leftarrow 0$

gera p

calcula o valor da adaptação dos cromossomos de p

enquanto (condição de parada não for satisfeita) **faça**

$t \leftarrow t + 1$

$\check{p} \leftarrow$ seleciona h cromossomos da população p

$\dot{p} \leftarrow$ aplica a operação de cruzamento e mutação sobre os cromossomos da população $\check{p}$

$p \leftarrow$ substitui l cromossomos da população p por l cromossomos da população $\dot{p}$

calcula o valor da adaptação dos cromossomos de p

fim do enquanto

Figura 2.1: Pseudocódigo de um algoritmo genético

A população dos cromossomos capazes de gerar descendente $\check{p}$ é construída pela seleção de h cromossomos da população p baseada na adaptação dos cromossomos ou em uma medida relativa ao mérito dos cromossomos dentro da população. Os cromossomos da população $\check{p}$ são submetidos à ação dos operadores genéticos de cruzamento e mutação para gerar descendente. A freqüência com que os cromossomos da população $\check{p}$ geram descendentes é estabelecida por uma medida de probabilidade associada a cada operação.

A nova população p é formada por l cromossomos da população $\dot{p}$ e por $n - l$ cromossomos da população p. O critério adotado para o nascimento dos l cromossomos da população $\dot{p}$ é o mesmo usado para a escolha dos cromossomos progenitores da população p, o qual é aplicado no sentido inverso para determinar os l cromossomos da população p que devem morrer. Por fim, é calculado o valor da adaptação dos cromossomos da população p recém construída.

Neste capítulo será apresentado um resumo das pesquisas sobre algoritmo genético realizadas até o momento, o qual, por motivo de organização, foi dividido em três seções. Na primeira seção serão apresentadas várias formas de representação para o cromossomo e a população; de inicializar a população e a probabilidade das operações de cruzamento e mutação; e de realizar as operações seleção, cruzamento, mutação e substituição, tanto para o algoritmo genético de Holland como para algumas variações deste. Na segunda seção serão comentados alguns modelos de algoritmos genéticos competentes, capazes de tornar um problema *intratável* para um algoritmo genético padrão em um problema *tratável* de modo rápido, confiável e preciso [55, 61].

Enquanto na última seção serão exibidos modelos de algoritmos genéticos *eficientes*, propostos para trabalhar problemas cujo ambiente, ao qual os cromossomos tem que se adaptar, é muito elaborado, estes algoritmos utilizam abordagens que estabelecem um equilíbrio entre a qualidade desejada e o custo computacional, em função do tempo e dos recursos consumidos [141, 143, 144].

2.2 Variações do Algoritmo Genético de Holland

2.2.1 Representação do Cromossomo

O cromossomo segundo Holland é um vetor binário de tamanho m, que armazena o resultado do problema de forma discretizada.

Definição 2.2.1. (Vetor Binário) - Um cromossomo é um vetor $\vec{i}_m^1 = \langle i_1, i_2, ..., i_m \rangle$, tal que, $\forall j \in \{1, 2, ..., m\}$ $(i_j = 0 \vee i_j = 1)$.

A adoção da representação do cromossomo por um vetor binário causa erros sempre que a quantidade de números binários que podem ser representados por um vetor de tamanho m é maior do que a cardinalidade do domínio do problema. Pensando em evitar estes erros alguns estudos sugerem a utilização de uma rotina de correção que descarta o cromossomo com erro [90]; ou que mapeia cromossomos com erro em cromossomos sem erros [13, 146].

Fox [51] adota a matriz *booleana* para representação do cromossomo ao tentar aplicar o algoritmo genético na solução de problemas, onde o relacionamento existente entre os elementos do domínio do problema deve ser considerado na construção do resultado, tal relacionamento é expresso através da associação das linhas e colunas desta matriz.

Definição 2.2.2. (Matriz Booleana) - Um cromossomo é uma matriz quadrática $\vec{i}_{j \times j}$, tal que, $\forall j_1, j_2 \in \{1, 2, ..., j\}$ $((i_{j_1,j_2} = 0 \implies$ *não existe um relacionamento entre o elemento do domínio* j_1 *e* $j_2) \wedge ((i_{j_1,j_2} = 1 \implies$ *existe um relacionamento entre o elemento do domínio* j_1 *e* $j_2))$.

Antonisse [4] propõe a representação do cromossomo por um vetor alfanumérico de comprimento fixo com e sem repetição, como uma forma de tornar a representação do cromossomo a mais natural possível para o problema em foco.

Definição 2.2.3. (Vetor Alfanumérico com Repetição) - Um cromossomo é um vetor $\vec{i}_1^m = \langle i_1, i_2, ..., i_m \rangle$, tal que, $(\forall j \in \{1, 2, ..., m\},\ i_j \in \{0, 1, ...9\} \cup \{a, b, ..., z\}) \wedge (\exists j_1, j_2 \in \{1, 2, ..., m\}\, i_{j_1} = i_{j_2})$.

Definição 2.2.4. (Vetor Alfanumérico sem Repetição) - Um cromossomo é um vetor $\vec{i}_1^m = \langle i_1, i_2, ..., i_m \rangle$, tal que, $(\forall j \in \{1, 2, ..., m\},\ i_j \in \{0, 1, ...9\} \cup \{a, b, ..., z\}) \wedge (\forall j_1, j_2 \in \{1, 2, ..., m\}\, i_{j_1} \neq i_{j_2})$.

Após alguns experimentos com estas representações Antonisse conclui que ela é incapaz de capturar a essência do resultado do problema, e passa a sugerir que o cromossomo seja representado por um vetor derivado de uma gramática [5].

Definição 2.2.5. (Vetor Derivado de Uma Gramática) - Um cromossomo derivado de uma gramática é um vetor $\vec{i}_1^m = \langle i_1, i_2, ..., i_m \rangle$, tal que existe uma gramática $\mathcal{G}$ cujas regras de produção derivam a palavra $i_1 i_2 ... i_m$.

Doravante o símbolo $\mathcal{C}_v$ será usado para representar o tipo cromossomo, cuja estrutura v é citada acima. Se v não for especificado significa que todos os tipos de representação para o cromossomo podem ser adotados.

2.2.2 Representação da População

A população segundo Holland é representada por um vetor de tamanho n, tal que cada linha desta matriz contém um cromossomo.

Definição 2.2.6. (Vetor) - Uma população é um vetor $\vec{p}_1^n = \langle p_1, p_2, ..., p_n \rangle$, tal que, $\forall j \in \{1, 2, ..., n\}\ p_j \in \mathcal{C}_v$.

Arabas, Mulawka e Pokrasniewicz [119] propõem a representação da população por uma lista de n-úplas contendo um cromossomo e dois valores inteiros, um para o parâmetro idade e outro para o parâmetro tempo de vida. Quando o cromossomo é gerado o valor do seu parâmetro idade é inicializado com o valor inteiro zero e cada vez que o laço **enquanto-faça** é executado, o valor do parâmetro idade é incrementado de uma unidade. O valor do seu parâmetro tempo de vida é inicializado por uma das seguintes funções:

- proporcional $vida(i) = \min(MinLT + \eta \times \frac{f(i)}{AvgFit}, MaxLT)$
- linear $vida(i) = MinLT + 2 \times \eta \times \frac{f(i) - |MinFit|}{|MaxFit| - |MinFit|}$

- bilinear $vida(i) = \begin{cases} MinLT + \eta \times \frac{f(i)-MinFit}{AvgFit-MinFit} & \text{se } AvgFit \geq f(i) \\ \frac{1}{2} \times (MinLT + MaxLT) + \eta \times \frac{f(i)-AvgFit}{MaxFit-AvgFit} & \text{se } AvgFit < f(i) \end{cases}$

onde $MinLT$ é o tempo de vida mínimo permitido, $MaxLT$ é o tempo de vida máximo permitido, η é o tempo de vida médio, $f(i)$ é o valor da adaptação do cromossomo i, $AvgFit$ é o valor da adaptação média da população, $MinFit$ é o valor da adaptação do cromossomo menos adaptado da população e $MaxFit$ é o valor da adaptação do cromossomo mais adaptado da população.

Definição 2.2.7. (Lista) - Uma população é uma lista $[i] = [(i_1, idade_1, vida_1), ..., (i_1, idade_n, vida_n)]$, tal que, $\forall j \in \{1, 2, ..., n\}\ i_j \in \mathcal{C}_v$.

Por conversão o símbolo $\mathcal{P}_{v_u}$ será usado para denotar o tipo população com representação v cujos cromossomos são representado pelo tipo u. Quando um dos índices não for declarado significa que a variável associada a ele pode apresentar qualquer uma das representações existentes.

2.2.3 Função de Adaptação

A adaptação de um cromossomo proposta por Holland é definida pela função objetivo e pela intenção do problema. A função objetivo do problema, representada por $\hat{o} : D \rightarrow \mathbb{R}$, recebe um elemento do domínio do problema D, e retorna a medida do grau de satisfação deste elemento ao problema. A intenção do problema informa se o resultado é o elemento com maior ou menor grau de satisfação. Assim, se a intenção do problema for encontrar o elemento com maior grau de satisfação, então a função de adaptação f do algoritmo genético será igual a $\hat{o}$, caso contrário a função de adaptação f do algoritmo genético será igual a $-\hat{o}$. Esta convenção se faz necessária porque o algoritmo genético, por definição, está sempre a procura do cromossomo mais adaptado.

Definição 2.2.8. (Adaptação) - A adaptação de um cromossomo é uma função f do seguinte tipo:

$f : \mathcal{C}_u \rightarrow \mathbb{R}$

$f(i) = \begin{cases} \hat{o}(decod(i)) & \text{se a intenção for encontrar o cromossomo com maior grau de satisfação} \\ -\hat{o}(decod(i)) & \text{se a intenção for encontrar o cromossomo com menor grau de satisfação} \end{cases}$

onde $decod$ é uma função que recebe um cromossomo e retorna o elemento do domínio do problema que este cromossomo representa.

Uma função de adaptação que reflete o valor "real" do cromossomo para o problema nem sempre serve para guiar a evolução da população do algoritmo genético para outra população mais adaptada. Por isto, algumas

metodologias capazes de auxiliar o usuário a construir uma função de adaptação adequada ao problema foram desenvolvidas. Entre as metodologias propostas estão os resultados da pesquisa de Cramer [32] que propõe a decomposição da função de adaptação em submetas; de Goldberg [58] que sugere uma função de adaptação cujo objetivo é penalizar os cromossomos que apresentam informações indesejáveis, ou que utiliza técnicas de aproximação de função para substituir a função de adaptação de um problema por uma versão simplificada da mesma; e de De Jong [94] que adota um método aceitável para a otimização de expressões lógicas *booleanas*. Destas metodologias, a mais difundida na comunidade de computação evolucionária é a baseada no conceito de penalidade $f_{pen} : \mathcal{C}_u \rightarrow \mathbb{R}$, cuja aplicação tornou-se comum depois da publicação do trabalho de Richardson [133] que descreve um conjunto de regras para se construir funções de adaptação segundo esta metodologia. Alguns exemplos de função de adaptação baseados no conceito de penalidade serão narrados a seguir.

Homaifar at all [81] propõem que f_{pen} seja determinada pela análise da família de intervalos que limitam cada restrição do problema do seguinte modo $f_{pen}(i) = f(i) + \sum_{j_1=1}^{w} \sum_{j_2=1}^{m} \mathcal{R}_{j_1,j_2}(i)$, onde f é a função de adaptação "real" do problema, w é o número de níveis de violação associado a cada posição do cromossomo, $\mathcal{R}_{j_1,j_2}(i)$ é o coeficiente de penalidade do nível de violação j_1 associado à posição j_2 do cromossomo i.

Joines e Houck [88] sugerem que f_{pen} seja calculada dinamicamente em função do número de iteração t pela seguinte formula $f_{pen}(i) = f(i) + (\gamma \times t)^{\alpha} \sum_{j=1}^{m} \mathcal{R}_j^{\beta}(i)$, onde f é a função de adaptação "real" do problema, $\mathcal{R}_j(i)$ é o coeficiente de penalidade de violação das restrições associadas à posição j do cromossomo i, e γ,α e β são constantes.

Powell e Skolnick [127] descrevem uma função de adaptação baseada na penalidade dinâmica e heurística dos cromossomos, tal que $f_{pen}(i) = f(i) + r_1 \times \sum_{j=1}^{m} \mathcal{R}_j(i) + \lambda(t, i)$, onde f é a função de adaptação "real" do problema, r_1 é uma constante, $\mathcal{R}_j(i)$ é o coeficiente de violação das restrições associadas à posição j do cromossomo i, e $\lambda(t, i)$ é uma função heurística que retorna zero para o cromossomo i que não viola qualquer restrição, caso contrário ela retorna um valor diferente de zero e crescente em função do parâmetro t.

2.2.4 Operador Genético de Seleção

A função de seleção elitista proposta por Holland recebe a população p e o parâmetro h e retorna uma população de tamanho h, formada pelos cromossomos mais adaptados da população p. Porém antes de apresentar a definição da operação de seleção elitista, será apresentada a definição da função filtro vetorial e ordenação vetorial utilizadas nas definições de operações que trabalham sobre objetos representados por vetores.

A função filtro vetorial recebe um vetor e dois números naturais j_1 e j_2, com $j_1 \in \{1, ..., w\}$ e $j_2 \in \{j_1 + 1, ..., w\}$, e retorna um vetor cujas informações contidas nas células das posições j_1,...,j_2 são cópias das informações contidas nestas mesmas posições do vetor fornecido e todas as outras posições são vazias, o que aqui será representado pelo símbolo $\vec{\sharp}$.

Definição 2.2.9. (Filtro Vetorial) - O filtro vetorial é uma função $\vec{\uparrow}$ do seguinte tipo:

$\vec{\uparrow} : \vec{V} \times \mathbb{N}^* \times \mathbb{N}^* \rightarrow \vec{V}$

$\vec{\uparrow}(\vec{x}_1^w, j_1, j_2) = \langle \vec{\sharp}, ..., \vec{\sharp}, x_{j_1}, x_{j_1+1}, ..., x_{j_2}, \vec{\sharp}, ..., \vec{\sharp} \rangle$

A função ordenação vetorial recebe um vetor e dois números naturais j_1 e j_2, com $j_1, j_2 \in \{1, ..., w + 1\}$, para indicar as posições que estão sendo comparadas em cada momento, e retorna um vetor cujas informações contidas nas suas células são as mesmas contidas nas células do cromossomo fornecido. Quando a função ordenação vetorial é chamada, o valor passado para o parâmetro j_1 e j_2 é "1".

Definição 2.2.10. (Ordenação Vetorial) - A ordenação vetorial é uma função $\vec{<}$ do seguinte tipo:

$\vec{<} : \vec{V} \times \mathbb{N}^* \times \mathbb{N}^* \rightarrow \vec{V}$

$$\vec{<}(\vec{x}_1^w, j_1, j_2) = \begin{cases} \vec{x}_1^w & \text{se } j_1 = w \text{ e } j_2 = w \\ \vec{<}(\vec{x}_1^w, j_1 + 1, 1) & \text{se } j_1 < w \text{ e } j_2 > w \\ \vec{<}(\vec{x}_1^w, j_1, j_2 + 1) & \text{se } j_1 < w, j_2 < w \text{ e } f(x_{j_1}) < f(x_{j_2}) \\ \vec{<}(\overline{\vec{x}_1^w}, j_1, j_2 + 1) & \text{se } j_1 < w, j_2 < w \text{ e } f(x_{j_1}) \geq f(x_{j_2}) \end{cases}$$

onde $\overline{\vec{x}_1^w} = \langle ..., x_{j_1-1}, x_{j_2}, x_{j_1+1}, ..., x_{j_2-1}, x_{j_1}, x_{j_2+1}, ... \rangle$

Definição 2.2.11. (Elitista) - A operação de seleção elitista é uma função *elitista* do seguinte tipo:

$elitista : \mathcal{P}_{Vetor_u} \times \mathbb{N}^* \rightarrow \mathcal{P}_{Vetor_u}$

$elitista(\vec{x}_1^n, h) = \vec{\uparrow}(\vec{<}(\vec{x}_1^n, 1, 1), n + 1 - h, n)$

Baker [8] aconselha a adoção da seleção de *rank* linear ou não, a qual se baseia na utilização de uma função definida pelo usuário para determinar a probabilidade do cromossomo que ocupa a posição j da população p estar presente na população $\check{p}$.

Definição 2.2.12. (*Rank* Linear) - A operação de seleção *rank* linear é uma função *rank* do seguinte tipo:

$rank : \mathcal{C}_v \rightarrow \mathbb{N}^*$

$rank(i_j) = r_1 \times f(i_j) + r_2$

onde r_1 e r_2 são constantes reais cujos valores não dependem do problema.

Definição 2.2.13. (*Rank* não Linear) - A operação de seleção *rank* não linear é uma função *power* do seguinte tipo:

$power : \mathcal{C}_v \rightarrow \mathbb{N}^*$

$power(i_j) = f(i_j)^{r_1}$

onde r_1 é uma constante real cujo valor depende do problema.

Goldberg [41] apresenta a seleção de *Boltzmann* que seleciona um cromossomo a partir de dois cromossomos i_1 e i_2 pela avaliação do grau de entropia da busca realizada pelo algoritmo genético.

Definição 2.2.14. (*Boltzmann*) - A operação de seleção *Boltzmann* é uma função *boltzmann* do seguinte tipo:

$boltzmann : \mathcal{C}_v \times \mathcal{C}_v \rightarrow \mathcal{C}_v$

$$boltzmann(i_1, i_2) = \begin{cases} i_1 & \text{se } \frac{1}{1+e^{\varrho}} \leq \frac{1}{2} \\ i_2 & \text{caso contrário} \end{cases}$$

onde $\varrho = \frac{f(i_1)-f(i_2)}{Temperatura}$.

De Jong [89] propõe a seleção de Monte Carlo, segundo a qual a relevância das informações codificadas em um cromossomo é diretamente proporcional ao valor da sua adaptação. Este operação começa com a construção de uma roleta, que nada mais é do que um vetor, cujas células só podem ser ocupadas pelos cromossomos da população p. O número de células da roleta, ocupadas por um cromossomo i da população p, é calculado pela seguinte fórmula: $roleta(i_1) = \frac{f(i_1)}{\sum_{i_2 \in p} f(i_2)} \times 360^o$. A seleção do cromossomo é feita gerando-se aleatoriamente um número natural j que pertence ao conjunto $\{1, ..., w\}$, onde w é o número de regiões da roleta. Se a região j da roleta for ocupada por um cromossomo, então este será o cromossomo selecionado, senão gera-se aleatoriamente um outro número j até que uma região ocupada seja obtida.

Definição 2.2.15. (Monte Carlo) - A operação de seleção de Monte Carlo é uma função *roleta* do seguinte tipo:

$roleta : \vec{V} \times \mathbb{N}^* \rightarrow \mathcal{C}_v$

$$roleta(\vec{v_1^w}, j) = \begin{cases} v_r & \text{se } v_j \neq \vec{\sharp} \\ roleta(\vec{v_1^w}, random(w)) & \text{caso contrário} \end{cases}$$

onde $\vec{\sharp}$ é o conteúdo de uma posição da roleta vazia e $random(w)$ é uma função de geração de números aleatórios pertencente ao intervalo $[1, w]$.

Brindle [21] sugere a seleção torneio, para permitir que a extinção das informações codificados nos cromossomos de baixa adaptação seja gradativa. A seleção torneio pode ser assim descrita: dado um conjunto de cromossomos da população atual p escolhidos aleatoriamente, selecione o cromossomo mais adaptado deste conjunto.

Definição 2.2.16. (Torneio) - A operação de seleção de torneio é uma função *torneio* do seguinte tipo:
$torneio : \mathcal{P}_{u_v} \times \mathbb{N}^* \rightarrow \mathcal{C}_v$
$torneio(x, w) = maisAdaptado(selecione(x, w))$
onde $maisAdaptado$ é a função que recebe um conjunto de cromossomo do tipo $\mathcal{C}_v$ e retorna um dos cromossomos mais adaptados do conjunto fornecido, e $seleciona$ é uma função que recebe uma população do tipo $\mathcal{P}_{u_v}$ e retorna um conjunto de cromossomos do tipo $\mathcal{C}_v$ com tamanho w formado por cromossomos da população fornecida escolhidos aleatoriamente.

2.2.5 Operador Genético de Cruzamento

A operação de cruzamento de Holland recebe um par de cromossomos da população $\breve{p}$, um ponto de corte j, com $j \in \{1, ..., m\}$, e um número aleatório r_1, com $r_1 \in [0, 100]$, e retorna o par de cromossomos fornecidos, se r_1 for maior que a probabilidade de cruzamento $prob_c$; caso contrário esta operação retorna um novo par de cromossomos formados pela combinação dos vetores dos cromossomos fornecidos. Porém antes de apresentar a definição da operação de cruzamento de Holland, será apresentada a definição da função adição vetorial utilizada nas definições das operações de cruzamento que trabalham sobre cromossomos representados por um vetor de números.

A função soma vetorial recebe dois vetores de mesma dimensão e um número natural j, para indicar a célula dos cromossomos fornecidos que está sendo avaliada em cada momento, e retorna um vetor cujas informações contidas nas suas células é o resultado de uma avaliação das informações existente nestas mesmas células nos vetores fornecidos. Quando a função soma vetorial é chamada, o valor passado para o parâmetro j é “1”.

Definição 2.2.17. (Soma vetorial) - A soma vetorial é uma função $\vec{+}$ do seguinte tipo:

$\vec{+} : \vec{V} \times \vec{V} \times \mathbb{N}^* \to \vec{V}$

$$\vec{+}(\vec{x}_1^w, \vec{y}_1^w, j) = \begin{cases} \vec{x}_1^w & \text{se } j > w \\ \vec{+}(\langle ..., x_{j-1}, x_j + y_j, x_{j+1}, ... \rangle, \vec{y}_1^w, j+1) & \text{se } j \leq w, x_j \neq \vec{\sharp} \text{ e } y_j \neq \vec{\sharp} \\ \vec{+}(\vec{x}_1^w, \vec{y}_1^w, j+1) & \text{se } j \leq w \text{ e } y_j = \vec{\sharp} \\ \vec{+}(\langle ..., x_{j-1}, y_j, x_{j+1}, ... \rangle, \vec{y}_1^w, j+1) & \text{se } j \leq w, x_j = \vec{\sharp} \text{ e } y_j \neq \vec{\sharp} \end{cases}$$

Definição 2.2.18. (Um Ponto de Corte) - A operação de cruzamento de um ponto de corte é uma função *um* do seguinte tipo:

$um : \mathcal{C}_v \times \mathcal{C}_v \times \mathbb{N}^* \times \mathbb{R}^+ \to \mathcal{C}_v \times \mathcal{C}_v$

$$um(\vec{x}_1^m, \vec{y}_1^m, j, r_1) = \begin{cases} (\vec{x}_1^m, \vec{y}_1^m) & \text{se } r_1 > prob_c \\ (\vec{+}(\vec{\uparrow}(\vec{x}_1^m, 1, j), \vec{\uparrow}(\vec{y}_1^m, j+1, m), 1), & \\ \vec{+}(\vec{\uparrow}(\vec{y}_1^m, 1, j), \vec{\uparrow}(\vec{x}_1^m, j+1, m), 1)) & \text{se } r_1 \leq prob_c \end{cases}$$

onde v pode ser um dos seguintes tipos de representação de cromossomo: vetor binário, vetor alfanumérico com e sem repetição.

Tag [159] propõe a operação de cruzamento de vários pontos de corte que recebe dois cromossomos, um conjunto de pontos de corte $\{j_1, ..., j_w\}$, com $j_1, ..., j_w \in \{1, ..., m\}$ e $j_1 < ... < j_w$, e um número aleatório r_1, com $r_1 \in [0, 100]$; e retorna dois cromossomos formados por aproximadamente $\frac{j_w+1}{2}$ pedaços alternados de cada um dos cromossomos fornecidos, se r_1 é menor ou igual à probabilidade de cruzamento, caso contrário ela retorna os cromossomos fornecidos.

Definição 2.2.19. (Vários Pontos de Corte) - A operação de cruzamento de vários pontos de corte é uma função *vários* do seguinte tipo:

vários: $\mathcal{C}_v \times \mathcal{C}_v \times \mathbb{P}(\mathbb{N}^+) \times \mathbb{R}^+ \to \mathcal{C}_v \times \mathcal{C}_v$

$$\textit{vários}(\vec{x}_1^m, \vec{y}_1^m, \{j_1, ..., j_w\}, r_1) = \begin{cases} (\vec{x}_1^m, \vec{y}_1^m) & \text{se } r_1 > prob_c \\ (\vec{+}(\vec{\uparrow}(\vec{x}_1^m, 1, j_1), (\vec{+}(\vec{\uparrow}(\vec{y}_1^m, j_1+1, j_2), ...)))), & \\ (\vec{+}(\vec{\uparrow}(\vec{y}_1^m, 1, j_1), (\vec{+}(\vec{\uparrow}(\vec{x}_1^m, j_1+1, j_2), ...)))) & \text{se } r_1 \leq prob_c \end{cases}$$

onde v pode ser um dos seguintes tipos de representação de cromossomo: vetor binário, vetor alfanumérico com e vetor alfanumérico sem repetição, e $\mathbb{P}(X)$ é o conjunto das partes do conjunto X.

Syswera [156] sugere a operação de cruzamento proporcional, baseada na constatação de que uma combinação rápida das informações codificadas nos cromossomos fornecidos para o cruzamento de vários pontos de corte pode ser obtida quando a distância entre os pontos de corte considerados for eqüitativa, ou seja, o conjunto de pontos de corte $\{j_1, ..., j_w\}$ é tal que: $j_1 = 1$; $j_w = m$; e $\sum_{a \in Impar} j_a - j_{a-1} = \sum_{b \in Par} j_b - j_{b-1}$, onde $Impar$ é o conjunto formado por todos os pontos de corte de índice ímpar com exceção de j_1 e j_w se w for ímpar, e Par é o conjunto formado por todos os pontos de corte de índice par com exceção de j_w se w for par.

Goldberg e Lingle [63] adotam o operador de cruzamento parcialmente mapeado, para ser aplicado a problemas de otimização combinatorial, que recebe dois cromossomos, uma relação $mapa$, que relaciona as informações contidas na mesma posição dos cromossomos fornecidos, dois pontos de corte j_1 e j_2, com $j_1 \in \{1, ..., m\}$ e $j_2 \in \{j_1 + 1, ..., m\}$, e um número aleatório r_1, com $r_1 \in [0, 100]$; e que retorna dois cromossomos formados pelas informações presentes em um dos cromossomos fornecidos nas posições j_1,...,j_2, e por preservar tanto quanto possível a ordem e as posições das informações contidas no outro cromossomo fornecido, se r_1 for menor ou igual à probabilidade de cruzamento, senão esta operação retorna os cromossomos fornecidos. Para tanto a especificação deste operador é dividida em duas fases. Na primeira fase dois cromossomos são formados pela permuta das informações compreendidas entre as posições j_1 e j_2 dos cromossomos fornecidos e na segunda as informações repetidas são removidas.

Definição 2.2.20. (Complemento) - A operação de permutação das informações contidas entres dois cromossomos submetidos ao operador de cruzamento parcialmene mapeado é realizada pela função *comp* do seguinte tipo:

$comp : \mathcal{C}_u \times \mathcal{C}_u \times \mathbb{N}^* \times \mathbb{N}^* \rightarrow \mathcal{C}_v \times \mathcal{C}_v$

$$\begin{aligned} comp(\vec{x}_1^m, \vec{y}_1^m, j_1, j_2) = \quad & (\vec{+}(\vec{\uparrow}(\vec{x}_1^m, 1, j_1 - 1), \vec{+}(\vec{\uparrow}(\vec{y}_1^m, j_1, j_2), \vec{\uparrow}(\vec{x}_1^m, j_2 + 1, m), 1), 1), \\ & \vec{+}(\vec{\uparrow}(\vec{y}_1^m, 1, j_1 - 1), \vec{+}(\vec{\uparrow}(\vec{x}_1^m, j_1, j_2), \vec{\uparrow}(\vec{y}_1^m, j_2 + 1, m), 1), 1)) \end{aligned}$$

onde u é o tipo vetor alfanumérico sem repetição e v é o tipo vetor alfanumérico com repetição.

Definição 2.2.21. (Remove) - A operação que remove as informações repetidas de um cromossomo submetido ao operador de cruzamento parcialmente mapeado é realizada pela função *rem* do seguinte tipo:

$rem : \mathcal{C}_v \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{P}(\mathcal{A} \times \mathcal{A}) \rightarrow \mathcal{C}_u$

$$rem(\vec{x_1}^m, j_1, j_2, j_3, map) = \begin{cases} \vec{x_1}^m & \text{se } j_3 > m \\ rem(\vec{x_1}^m, j_1, j_2, j_3 + 1, map) & \text{se } j_3 \leq m \text{ e, } j_3 \in \{j_1, ..., j_2\} \text{ ou} \\ & j_3 \notin \{j_1, ..., j_2\} \wedge \psi(x_{j_3}, \vec{x_1}^m) = 1 \\ rem(\overline{\vec{x_1}^m}, j_1, j_2, j_3 + 1, map) & \text{se } j_3 \leq m, \, j_3 \notin \{j_1, ..., j_2\} \text{ e } \psi(x_{j_3}, \vec{x_1}^m) > 1 \end{cases}$$

onde u é o tipo vetor alfanumérico sem repetição, v é o tipo vetor alfanumérico com repetição, $\mathcal{A}$ é o conjunto de símbolos alfanuméricos, $\overline{\vec{x_1}^m} = \langle x_1, ..., x_{j_3-1}, map.2(x_{j_3}), x_{j_3+1}, ..., x_m \rangle$ com $map.2(x_{j_3})$ sendo a informação contida no segundo parâmetro da relação map cuja informação do primeiro parâmetro é x_{j_3}.

Definição 2.2.22. (Parcialmente mapeado) - A operação de cruzamento parcialmente mapeado é uma função *parcial* do seguinte tipo:

$parcial : \mathcal{C}_v \times \mathcal{C}_v \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{R}^+ \times \mathbb{P}(\mathcal{A} \times \mathcal{A}) \rightarrow \mathcal{C}_v \times \mathcal{C}_v$

$$parcial(\vec{x_1}^m, \vec{y_1}^m, j_1, j_2, r_1, mapa) = \begin{cases} (\vec{x_1}^m, \vec{y_1}^m) & \text{se } r_1 > prob_c \\ (rem(comp(\vec{x_1}^m, \vec{y_1}^m, j_1, j_2), j_1, j_2, 1, mapa), & \\ rem(comp(\vec{y_1}^m, \vec{x_1}^m, j_1, j_2), j_1, j_2, 1, \overline{mapa})) & \text{se } r_1 \leq prob_c \end{cases}$$

onde v é o tipo vetor alfanumérico sem repetição, $\mathcal{A}$ é o conjunto de símbolos alfanuméricos e $\overline{mapa}$ é uma relação tal que para todo elemento $(a_1, a_2) \in mapa$ existe um elemento $(a_2, a_1) \in \overline{mapa}$.

Davis [36] apresenta o cruzamento ordenado, para ser aplicado a problemas de otimização combinatorial, que recebe dois cromossomos, dois pontos de corte j_1 e j_2, com $j_1 \in \{1, ..., m\}$ e $j_2 \in \{j_1 + 1, ..., m\}$, e um número aleatório r_1, com $r_1 \in [0, 100]$, e retorna dois cromossomos formados pelas informações constantes nas posições $j_1, ..., j_2$ em um dos cromossomos fornecidos preservando a ordem em que as informações não constantes nestas posições ocorrem no outro cromossomo, se r_1 for menor ou igual à probabilidade de cruzamento, senão retorna os cromossomos fornecidos. Tal preservação será realizada pela função $pres$ que recebe dois cromossomos, o cromossomo em formação e o cromossomo cuja ordem de suas informações deve ser preservada, e duas variáveis para contar as posições avaliadas nestes cromossomos em cada momento.

Definição 2.2.23. (Preenchimento) - A operação de preenchimento das informações para o cruzamento ordenado é uma função *pres* do seguinte tipo:

$pres : \mathcal{C}_v \times \mathcal{C}_u \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}^* \rightarrow \mathcal{C}_u$

$$pres(\vec{x}_1^m, \vec{y}_1^m, j_1, j_2, j_3, j_4) = \begin{cases} \vec{x}_1^m & \text{se } j_3 > m \\ pres(\vec{x}_1^m, \vec{y}_1^m, j_1, j_2, j_3 + 1, j_4) & \text{se } j_3 \leq m, j_3 \notin \{j_1, ..., j_2\} \text{ e} \\ & \psi(x_{j_3}, \vec{x}_1^m) = 1 \\ pres(\vec{x}_1^m, \vec{y}_1^m, j_1, j_2, j_3, j_4 + 1) & \text{se } j_3 \leq m, j_3 \notin \{j_1, ..., j_2\} \text{ e} \\ & \psi(y_{j_4}, \vec{x}_1^m) = 1 \\ pres(\overline{\vec{x}_1^m}, \vec{y}_1^m, j_1, j_2, j_3 + 1, j_4 + 1) & \text{se } j_3 \leq m, j_3 \notin \{j_1, ..., j_2\} \text{ e} \\ & \psi(y_{j_4}, \vec{x}_1^m) = 0 \end{cases}$$

onde u é o tipo vetor alfanumérico sem repetição, v vetor alfanumérico com repetição, $\overline{\vec{x}_1^m} = \langle ..., x_{j_3-1}, y_{j_4}, x_{j_3+1}, ... \rangle$.

Definição 2.2.24. (Ordenado) - A operação de cruzamento ordenado é uma função *ordem* do seguinte tipo:

$ordem : \mathcal{C}_v \times \mathcal{C}_v \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{R}^+ \rightarrow \mathcal{C}_v \times \mathcal{C}_v$

$$ordem(\vec{x}_1^m, \vec{y}_1^m, j_1, j_2, r_1) = \begin{cases} (\vec{x}_1^m, \vec{y}_1^m) & \text{se } r_1 > prob_c \\ (pres(comp(\vec{x}_1^m, \vec{y}_1^m, j_1, j_2), j_1, j_2, 1, 1)), & \\ pres(comp(\vec{y}_1^m, \vec{x}_1^m, j_1, j_2), j_1, j_2, 1, 1))) & \text{se } r_1 \leq prob_c \end{cases}$$

onde v é o tipo vetor alfanumérico sem repetição.

Antonisse [5] utiliza o cruzamento por derivação, para ser aplicado sobre cromossomos representados por vetores construídos a partir de uma gramática, que recebe dois cromossomos $\vec{x}_1^{m_1}$ e $\vec{y}_1^{m_2}$, dois pontos de corte j_1 e j_2, com $j_1 \in \{1, ..., m_1\}$ e $j_2 \in \{1, ..., m_2\}$, e um número aleatório r_1, com $r_1 \in [0, 100]$, e retorna dois cromossomos formados pelas informações constantes em um dos cromossomos fornecidos, se r_1 for menor ou igual à probabilidade de cruzamento e a regra de produção da gramática que derivou o símbolo x_{j_1} no vetor $\vec{x}$ for à mesma que derivou o símbolo y_{j_2} no vetor $\vec{y}$, senão retorna os cromossomos fornecidos.

Definição 2.2.25. (Derivação) - A operação de cruzamento por derivação é uma função *deriva* do seguinte tipo:

$deriva : \mathcal{C}_v \times \mathcal{C}_v \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{R}^+ \rightarrow \mathcal{C}_v \times \mathcal{C}_v$

$$ordem(\vec{x}_1^{m_1}, \vec{y}_1^{m_2}, j_1, j_2, r_1) = \begin{cases} (\vec{x}_1^{m_1}, \vec{y}_1^{m_2}) & \text{se } \mathcal{G}(j_1, \vec{x}_1^{m_1}) \neq \mathcal{G}(j_2, \vec{y}_1^{m_2}) \text{ ou } r_1 > probc \\ (\widetilde{\vec{x}_1^{m_1}}, \widetilde{\vec{y}_1^{m_2}}) & \text{se } \mathcal{G}(j_1, \vec{x}_1^{m_1}) = \mathcal{G}(j_2, \vec{y}_1^{m_2}) \text{ e } r_1 \leq prob_c \end{cases}$$

onde v é o tipo vetor derivado de uma gramática, $\mathcal{G}(z, \vec{z})$ retorna a regra de produção da gramática $\mathcal{G}$ que derivou o símbolo z no vetor $\vec{z}$, $\widetilde{\vec{x}_1^{m_1}} = \langle ..., x_{j_1-1}, y_{j_2}, ... \rangle$ e $\widetilde{\vec{y}_1^{m_2}} = \langle ..., y_{j_2-1}, x_{j_1}, ... \rangle$.

Fox [51] propõe o cruzamento por conjunção e disjunção, para ser aplicado sobre cromossomos representados por matrizes *booleanas*. O cruzamento por conjunção recebe dois cromossomos e um número aleatório r_1,

com $r_1 \in [0, 100]$, e retorna um cromossomo formado pela conjunção das informações constantes nos cromossomos fornecidos, se r_1 for menor ou igual à probabilidade de cruzamento, senão retorna um dos cromossomos fornecidos.

Definição 2.2.26. (Conjunção) - A operação de cruzamento por conjunção é uma função *and* do seguinte tipo:

$$and : \mathcal{C}_v \times \mathcal{C}_v \times \mathbb{R}^+ \rightarrow \mathcal{C}_v$$

$$and(\vec{x}_{m\times m}, \vec{y}_{m\times m}, r_1) = \begin{cases} \vec{x}_{m\times m} & \text{se } r_1 > probc \\ \langle x_{1,1} \widehat{+} y_{1,1}, ..., x_{m,m} \widehat{+} y_{m,m} \rangle & \text{se } r_1 \leq prob_c \end{cases}$$

onde v é o tipo matriz *booleana* e $a \widehat{+} b$ é uma função que retorna "1" se a e b forem "1", caso contrário ela retorna "0".

O cruzamento por disjunção recebe dois cromossomos e um número aleatório r_1, com $r_1 \in [0, 100]$, e retorna um cromossomo formado pela disjunção das informações constantes nos cromossomos fornecidos, se r_1 for menor ou igual à probabilidade de cruzamento, senão retorna um dos cromossomos fornecidos.

Definição 2.2.27. (Disjunção) - A operação de cruzamento por disjunção é uma função *or* do seguinte tipo:

$$or : \mathcal{C}_v \times \mathcal{C}_v \times \mathbb{R}^+ \rightarrow \mathcal{C}_v$$

$$or(\vec{x}_{m\times m}, \vec{y}_{m\times m}, r_1) = \begin{cases} \vec{x}_{m\times m} & \text{se } r_1 > probc \\ \langle x_{1,1} \widehat{\times} y_{1,1}, ..., x_{m,m} \widehat{\times} y_{m,m} \rangle & \text{se } r_1 \leq prob_c \end{cases}$$

onde v é o tipo matriz *booleana* e $a \widehat{\times} b$ é uma função que retorna "1" se a ou b forem "1", caso contrário ela retorna "0".

2.2.6 Operador Genético de Mutação

Holland propõe duas operações de mutação: por complemento e por inversão. A mutação por complemento recebe um cromossomo, um conjunto de posições $\{j_1, ..., j_w\}$, com $\{j_1, ..., j_w\} \subseteq \{1, ..., m\}$ e $j_1 < \ldots < j_w$, e um número aleatório r_1, com $r_1 \in [0, 100]$; e retorna o cromossomo fornecido, se r_1 é maior que a probabilidade de mutação $prob_m$, caso contrário um novo cromossomo cujas informações apresentadas nas posições $j_1, ..., j_w$ são diferentes das apresentadas no cromossomo fornecido.

Definição 2.2.28. (Complemento) - A operação de mutação por complemento de Holland é uma função *complemento* do seguinte tipo:

$complemento : C_v \times \mathbb{P}(\mathbb{N}^*) \times \mathbb{R}^+ \rightarrow \mathcal{C}_v$

$$complemento(\vec{x}_1^m, \{1_1, ..., j_w\}, r_1) = \begin{cases} \vec{x}_1^m & \text{se } r_1 > prob_m, \\ \langle x_1, ..., x_{j_1-1}, \bar{x}_{j_1}, x_{j_1+1}, ..., x_{j_2-1}, \bar{x}_{j_2}, ..., x_m \rangle & \text{se } r_1 \leq prob_m. \end{cases}$$

onde v é o tipo vetor binário, e $\bar{x}_j$ é "0" se $x_j = 1$, caso contrário é "1".

A mutação por inversão recebe um cromossomo, dois pontos de corte j_1 e j_2, com $j_1 \in \{1, ..., m\}$ e $j_2 \in \{j_1 + 1, ..., m\}$, e um número aleatório r_1, com $r_1 \in [0, 100]$, e retorna o cromossomo fornecido, se r_1 for maior do que a probabilidade de mutação, caso contrário um novo cromossomo formado pela inversão das informações que ocupam as posições $j_1, ..., j_2$ do cromossomo fornecido.

Definição 2.2.29. (Inversão) - A operação de mutação por inversão é uma função *inverte* do seguinte tipo:

$inverte : \mathcal{C}_v \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{R}^+ \rightarrow \mathcal{C}_v$

$$inverte(\vec{x}_1^m, j_1, j_2, r_1) = \begin{cases} \vec{x}_1^m & \text{se } r_1 > prob_m, \\ \langle x_1, ..., x_{j_1-1}, x_{j_2}, ..., x_{j_1}, x_{j_2+1}, ..., x_m \rangle & \text{se } r_1 \leq prob_m. \end{cases}$$

onde v é o tipo vetor binário, ou vetor alfanumérico com repetição ou vetor alfanumérico sem repetição.

Hsu et al [83] sugerem a mutação por troca que recebe um cromossomo, um conjunto de pontos de troca de cardinalidade par $\{j_1, ..., j_w\}$, com $j_1, ..., j_w \in \{1, ..., m\}$ e $j_1 < ... < j_w$, e um número aleatório r_1, com $r_1 \in [0, 100]$; e retorna o cromossomo fornecido, se r_1 for maior do que a probabilidade de mutação, caso contrário retorna um novo cromossomo com as informações contidas nas posições j_w e j_{w+1} trocadas, para $w \in \{1, 3, 5, ...\}$.

Definição 2.2.30. (Troca) - A operação de mutação por troca é uma função *troca* do seguinte tipo:

$troca : C_v \times \mathbb{P}(\mathbb{N}^*) \times \mathbb{R}^+ \rightarrow \mathcal{C}_v$

$$troca(\vec{x}_1^m, \{j_1, ..., j_w\}, r_1) = \begin{cases} \vec{x}_1^m & \text{se } r_1 > prob_m, \\ \langle ..., x_{j_1-1}, x_{j_2}, x_{j_1+1}, ..., x_{j_2-1}, x_{j_1}, x_{j_2+1}, ..., x_m \rangle & \text{se } r_1 \leq prob_m. \end{cases}$$

onde v é o tipo vetor binário, ou o tipo vetor alfanumérico com repetição ou vetor alfanumérico sem repetição.

Fox [51] propõe a mutação por negação, para ser aplicada sobre cromossomos representados por matrizes *booleanas*, que recebe um cromossomo e um número aleatório r_1, com $r_1 \in [0, 100]$, e retorna um cromossomo formado pela negação das informações constantes no cromossomo fornecido, se r_1 for menor ou igual à probabilidade de mutação, senão retorna os cromossomos fornecidos.

Definição 2.2.31. (Negação) - A operação de cruzamento por negação é uma função *not* do seguinte tipo:

$not : \mathcal{C}_v \times \mathbb{R}^+ \rightarrow \mathcal{C}_v$

$$not(\vec{x}_{m\times m}, r_1) = \begin{cases} \vec{x}_{m\times m} & \text{se } r_1 > probm \\ \langle \widehat{-}x_{1,1}, ..., \widehat{-}x_{m,m} \rangle & \text{se } r_1 \leq prob_c \end{cases}$$

onde v é o tipo matriz *booleana* e $\widehat{-}a$ é uma função que retorna "1" se a for "0", caso contrário ela retorna "0".

2.2.7 Operador Genético de Substituição

A operação de substituição elitista proposta por Holland recebe a população atual p, a população dos cromossomos novos $\dot{p}$ e o parâmetro l, e retorna uma nova população formada pelos l cromossomos mais adaptados da população $\dot{p}$ e pelos $n - l$ cromossomos mais adaptados da população p. A construção de um novo vetor a partir das fornecidas contidas em dois outros vetores de diferentes dimensões é realizada pela função concatenação vetorial.

Definição 2.2.32. (Concatenação Vetorial) - A concatenção vetorial é uma função $\vec{\frown}$ do seguinte tipo:

$\vec{\frown} : \mathcal{P}_{u_v} \times \mathcal{P}_{u_v} \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}^* \rightarrow \mathcal{P}_{u_v}$

$\vec{\frown}(\vec{x}_{n_1\times m}, \vec{y}_{n_2\times m}, j_1, j_2, j_3, j_4) = \langle x_{j_1}, ..., x_{j_2}, y_{j_3}, ..., y_{j_4} \rangle$

Definição 2.2.33. (Elitista) - A operação de substituição elitista é uma função *substitui* do seguinte tipo:

$substitui : \mathcal{P}_{u_v} \times \mathcal{P}_{u_v} \times \mathbb{N}^* \rightarrow \mathcal{P}_{u_v}$

$substitui(\vec{p}_{n\times m}, \vec{\dot{p}}_{w\times m}, l) = \vec{\frown}(\vec{<}(\vec{p}_{n\times m}), \vec{<}(\vec{\dot{p}}_{w\times m}), l+1, n, w+1-l, w)$

Mahfound [112] propõe a operação de substituição determinística que recebe a população atual p, a população dos cromossomos novos $\dot{p}$, uma população auxiliar da mesma dimensão da população p, o parâmetro l, e dois números naturais j_1 e j_2, para indicar as posições que estão sendo comparadas da população p e $\dot{p}$ respectivamente; e retorna uma nova população da mesma cardinalidade da população p formada pelos cromossomos da população p que não conseguiram gerar por cruzamento ou mutação nenhum descendente mais adaptado, ou pelos cromossomos da população $\dot{p}$ que forem mais adaptados do que os cromossomos que lhes deram origem. Quando a mutação determinística é inicializada j_1 e j_2 recebem o valor "1", e a população auxiliar recebe o valor "$\vec{\sharp}$", símbolo utilizado para denotar as posições vazias de uma população.

Definição 2.2.34. (Determinística) - A operação de substituição determinística é uma função *der* do seguinte tipo:

$$der : \mathcal{P}_{u_v} \times \mathcal{P}_{u_v} \times \mathcal{P}_{u_v} \times \mathbb{N}^* \times \mathbb{N}^* \rightarrow \mathcal{P}_{u_v}$$

$$der(p, \dot{p}, x, j_1, j_2) = \begin{cases} x & \text{se } j_1 > m \\ der(p, \dot{p}, x, j_1 + 1, 1) & \text{se } j_1 \leq m \text{ e } j_2 > m \\ der(p, \dot{p}, \ddot{x}, j_1, j_2 + 1) & \text{se } j_1 \leq m \text{ e } j_2 \leq m \end{cases}$$

onde $\ddot{x}$ é a população com a posição j_1 preenchida com o cromossomo da população p que ocupa a posição j_1, ou com um dos cromossomos resultantes da mutação deste cromossomo existentes na população $\dot{p}$ que apresenta valor de adaptação maior do que $f(p_{j_1})$, ou com um dos cromossomos gerados pela aplicação da operação de cruzamento sobre os cromossomos que ocupam a posição j_1 e j_2 da população p existentes na população $\dot{p}$ que apresenta valor de adaptação maior do que $f(p_{j_1})$.

Arabas, Mulawka e Pokrasniewicz [6, 119] sugerem a substituição por tempo de vida, que utiliza a representação da população proposta pelos mesmos, descrita na Seção 2.2.2. A substituição por tempo de vida forma uma nova população x com todos os cromossomos da população atual p e auxiliar $\dot{p}$, cujo valor do seu parâmetro idade é menor ou igual ao valor do seu parâmetro tempo de vida. Quando esta operação é chamada o parâmetro população x deve ser passado como uma lista vazia, representada aqui pelo símbolo $[]$.

Definição 2.2.35. (Por Tempo de Vida) - A operação de substituição por tempo de vida é uma função *vida* do seguinte tipo:

$$vida : \mathcal{P}_{u_v} \times \mathcal{P}_{u_v} \rightarrow \mathcal{P}_{u_v}$$

$$vida([p], [\dot{p}], [x]) = \begin{cases} x & \text{se } [p] = [] \text{ e } [\dot{p}] = [] \\ vida(tail_{[p]}, [\dot{p}], head_{[p]} : [x]) & \text{se } [p] \neq [] \text{ e } head_{[p]}.idade \leq head_{[p]}.vida \\ vida(tail_{[p]}, [\dot{p}], [x]) & \text{se } [p] \neq [] \text{ e } head_{[p]}.idade > head_{[p]}.vida \\ vida([p], tail_{[\dot{p}]}, head_{[\dot{p}]} : [x]) & \text{se } [p] = [], [\dot{p}] \neq [] \text{ e } head_{[\dot{p}]}.idade \leq head_{[p]}.vida \\ vida([p], tail_{[\dot{p}]}, [x]) & \text{se } [p] = [], [\dot{p}] \neq [] \text{ e } head_{[\dot{p}]}.idade > head_{[p]}.vida \end{cases}$$

onde u é o tipo lista, $head_a$ é a função que retorna o primeiro elemento da lista $[a]$, $tail_a$ é a função que retorna a lista $[a]$ sem o primeiro elemento, $a : [b]$ é a função insere o elemento a na cabeça da lista $[b]$, $g.idade$ é o valor do parâmetro idade do nó g de uma população do tipo lista, e $g.vida$ é o valor do parâmetro tempo de vida do nó g de uma população do tipo lista.

2.2.8 Probabilidade de Cruzamento e Mutação

A freqüência com que os operadores genéticos de cruzamento e mutação são aplicados sobre os cromossomos da população atual pode acelerar ou retardar a convergência da população manipulada pelo algoritmo genético. Sendo a definição do valor da probabilidade de cruzamento e mutação uma dos tópicos investigados por alguns pesquisadores [16, 20, 36, 37, 38, 85, 156, 14].

Uma das primeiras abordagens apresentadas para atribuir dinamicamente a probabilidade dos operadores genéticos de cruzamento e mutação foi variar linearmente $prob_c$ e $prob_m$ durante a execução do algoritmo genético, tal que a cada iteração $prob_c$ é decrementada de uma quantidade r_1 e $prob_m$ é incrementada de uma quantidade r_2 [36, 156], com $r_1, r_2 \in \mathbb{R}^+$. Booker [16] sugere que somente a probabilidade de cruzamento deve ter seu valor atribuído dinamicamente em função da adaptação média da população.

Outra abordagem de atribuição dinâmica para a probabilidade de cruzamento e mutação foi proposta por Davis [37, 38]. Segundo a qual a probabilidade de cada operador recebe um incremento de uma quantidade r_1, $r_1 \in \mathbb{R}^+$, toda vez que sua ação gera um cromossomo mais adaptado do que algum cromossomo da população atual, e um decremento de uma quantidade r_1 caso contrário. Existe ainda um conjunto de trabalhos baseados na atribuição dinâmica da operação de mutação, que é decrementada por um fator exponencial durante a execução do algoritmo genético [20, 85].

2.3 Algoritmos Genéticos Competentes

2.3.1 Algoritmo de Otimização *bayesiana*

Um algoritmo de otimização *bayesiana* [122, 123, 124, 121, 96] utiliza o conhecimento sobre o problema acumulado em redes *bayesianas* para conduzir a evolução das populações manipuladas pelo algoritmo genético (Figura 2.2). Nestes algoritmos o cromossomo e a população têm a mesma representação proposta por Holland. O mecanismo de seleção pode ser qualquer um dos apresentados na Seção 2.2.4. A função de adaptação é uma medida calculada sobre a rede *bayesiana* construída a partir das informações codificadas nos cromossomos selecionados. A única operação de geração de cromossomos utilizada é o cruzamento por junção de acordo com a rede *bayesiana* construída. O mecanismo de substituição pode ser qualquer um dos descritos na Seção 2.2.7.

Uma rede *bayesiana* $\mathcal{B}$ é um grafo acíclico dirigido com os nodos correspondendo às variáveis do conjunto de dados modelado. O cruzamento por junção é uma distribuição probabilística dada por $\delta(\vec{x}_1^m) = \prod_{j=1}^{m} \delta(x_j \mid$

$t \leftarrow 0$

gera p

repita

selecione uma população $\check{p}$ de cromossomos da população p

construa uma rede *bayesiana* $\mathcal{B}$ usando uma medida de qualidade e restrição sobre $\check{p}$

gere uma população de novos cromossomos $\dot{p}$ com a distribuição de junção codificada em $\mathcal{B}$

substitua alguns cromossomos da população p pelos cromossomos da população $\dot{p}$

$t \leftarrow t + 1$

até (condição de parada ser satisfeita)

Figura 2.2: Algoritmo de otimização *bayesiana*

$\prod_{x_j}$), onde $\vec{x}_1^m$ é o cromossomo, $\prod_{x_j}$ é o conjunto dos nós de $\mathcal{B}$ que figuram como ponto de partida para os arcos cujo destino é o nó x_j e $\delta(x_j \mid \prod_{x_j})$ é a probabilidade condicional de x_j dado $\prod_{x_j}$. A direção do arco (a, b), que sai do nó a e chega ao nó b, em uma rede *bayesiana* indica que a variável associada ao nó b é condicionada pela variável relacionada ao nó a. Se mais de um arco chegam a um dado nó a, então a probabilidade da variável associada ao nó a é o resultado de uma probabilidade condicionada desta variável com a probabilidade de junção condicionada de todas as variáveis de $\prod_a$.

A construção da rede *bayesiana* sobre o conjunto de cromossomos selecionados pode ser feita através de vários métodos, todos baseados em uma medida da estimativa do grau de crença garantido em função as evidências, que discrimina a rede de acordo com a sua qualidade, e um algoritmo de busca, que caminha sobre a rede *bayesiana* para desenhar um caminho com a melhor estimativa do grau de crença. Uma das medidas de estimativa do grau de crença mais utilizada é a *bayesiana-Dirichlet* [78] que combina o conhecimento a *priori* sobre o problema e os dados estatísticos obtidos a partir do conjunto de dados fornecidos.

A probabilidade de uma rede *bayesiana* $\mathcal{B}$ dada a variável x_j pode ser calculada pela fórmula $\delta(\mathcal{B} \mid x_j) = \frac{\delta(\mathcal{B}) \times \delta(x_j|\mathcal{B})}{\delta x_j}$, onde $\delta(\mathcal{B})$ é a probabilidade a priori da rede *bayesiana* $\mathcal{B}$ ser verdadeira na ausência de qualquer evidência específica, $\delta(x_j \mid \mathcal{B})$ é a probabilidade da evidência x_j ser verdadeira dada a rede $\mathcal{B}$, e $\delta(x_j)$ é a probabilidade a priori da evidência x_j ser verdadeira na ausência de qualquer hipótese. Entre os algoritmos de busca os que mais se destacam são *hillclimbing* [118, 123] e *simulated annealing* [99, 54].

2.3.2 Algoritmo Genético *Messy* com Gene Expresso

Os algoritmos genéticos baseados no trabalho de Holland geram uma competição entre a identificação de blocos de construção, realizada pela operação de seleção, e a combinação de blocos de construção, realizada pela operação de cruzamento. Sendo esta competição responsável pela dificuldade encontrada pelo algoritmo genético de Holland e suas variações de encontrar um resultado aceitável para problemas com *epistasis* na ausência de informações sobre a interação existente entre os blocos de construções da população atual. Uma forma de contornar tal dificuldade é separar estas operações em fases como proposto pelo o algoritmo genético *messy* com gene expresso [95, 29] (ver Figura 2.3).

O algoritmo genético *messy* com gene expresso é uma extensão do algoritmo genético *messy* [62] que incorpora a idéia de gene expresso da genética, para pesquisar diretamente as relações existentes entre os cromossomos gerados, tarefa esta realizada pela operação de *TranscriçãoI* executada durante a fase primordial e *TranscriçãoII* executada antes do início da fase de justaposição. Neste algoritmo a população é representada por um conjunto de cromossomos, que por sua vez é representado por um vetor de n-uplas $(locus, valor, peso, linkage)$ denominadas de genes, onde $locus$ determina a variável do problema que é otimizada pelo gene, $valor$ determina o elemento do alfabeto atribuído à variável $locus$, $peso$ identifica a classificação do gene, e $linkage$ é uma lista de variáveis relacionadas com a variável $locus$ do gene em análise.

A figura abaixo apresenta um pseudo-código para o algoritmo genético *messy* com gene expresso. As variáveis $w_{máximo}$, m e $t_{máximo}$ desta figura correspondem ao tamanho do alfabeto, ao tamanho do cromossomo e ao número máximo de iterações permitidas, respectivamente.

A operação *TranscriçãoI* determina as instâncias de genes que devem contribuir para um ótimo local, pela alteração do parâmetro $peso$ dos genes de um cromossomo de modo que, se existir algum elemento do alfabeto cuja atribuição para o parâmetro $valor$ de um gene contribuir para melhorar a adaptação do cromossomo que o contém, então o parâmetro $peso$ deste gene deverá receber o valor da diferença entre a adaptação do cromossomo com o gene alterado e com o gene original, senão o $peso$ do gene será “0”. Enquanto a operação *TranscriçãoII* determina precisamente a relação entre os genes que conduzem a um resultado ótimo local, pela inserção de um gene que possui $peso$ maior do que “0” no parâmetro $linkage$ de outro gene que também apresenta $peso$ maior do que “0”, desde que uma perturbação no parâmetro $valor$ de um destes genes produza um decremento no valor da adaptação do cromossomo a que eles pertencem.

gera p
\\ fase primordial
para $w \in [0, w_{máximo}[$ **faça**
    para $j \in [0, m[$ **faça**
        TranscriçãoI(p,j)
    fim do para
fim do para
TranscriçãoII(p,j)
\\ fase de justaposição
para $t \in [0, t_{máximo}]$ **faça**
    Seleção (p)
    SeleçãoDeClasse(p)
    Cruzamento(p)
    calcula a adaptação de p
fim do para

Figura 2.3: Algoritmo genético *messy* com gene expresso

O algoritmo genético *messy* com gene expresso utiliza duas operações de seleção: uma para escolher os cromossomos que serão submetidos ao cruzamento, seleção torneio (Definição 2.2.16), e outra para escolher os genes pertencentes a dois cromossomos, submetidos ao cruzamento, que apresentam o maior grau de dependência entre si.

A operação de cruzamento do algoritmo genético *messy* com gene expresso recebe dois cromossomos e para cada gene destes cromossomos gera um número aleatório r_1, se r_1 for menor ou igual à probabilidade de cruzamento e o parâmetro $peso$ do primeiro cromossomo for maior do que o do segundo cromossomo, então os parâmetros $linkage$ dos genes fornecidos são trocados. Os cromossomos retornados por esta operação de cruzamento irão substituir os cromossomos da população p que lhes deram origem.

2.3.3 Algoritmo Genético Baseado em Aprendizagem por *Linkage*

O algoritmo genético baseado em aprendizagem por *linkage* [74] propõe que o cromossomo representando por um vetor $\vec{x}_1^m$ binário seja modelado internamente por uma lista circular $[x]$ de pares (j, x), onde j é uma

célula do vetor, e x é o elemento do alfabeto que ocupa a célula j do cromossomo $\vec{x}_1^m$. A lista $[x]$ pode conter mais de um nó com o mesmo valor j, sendo que somente um destes elementos irá contribuir efetivamente para a adaptação do cromossomo $[x]$. Os pares da lista $[x]$ que não contribuem para a sua adaptação são chamados de *introns*. A presença de *introns* no cromossomo $[x]$ é usada para facilitar a propagação de blocos de construção e a formação de *linkage*. Associado a cada cromossomo $[x]$ existe um ponto de interpretação, que indica a partir de que posição da lista o cromossomo deve começar a sua decodificação para a representação vetorial, nesta decodificação a primeira ocorrência de cada posição j é considerada e as outras desconsideradas. Assim dependendo do ponto de interpretação escolhido um mesmo cromossomo $[x]$ pode codificar diferentes cromossomos $\vec{x}$.

No algoritmo genético baseado em aprendizagem por *linkage* a única operação de construção de novos cromossomos usada é o cruzamento que recebe dois cromossomos $[x]$ e $[y]$, denotados respectivamente de cromossomo receptor e cromossomo doador, três pontos de corte j_1, j_2 e j_3, com $1 \leq j_1 \leq tam_{[x]}$, $1 \leq j_2 \leq tam_{[y]}$ e $j_2 \leq j_3 \leq tam_{[y]}$, onde $tam_{[x]}$ é o tamanho da lista $[x]$ e $tam_{[y]}$ é o tamanho da lista $[y]$, e um número aleatório r_1, e retorna os cromossomos fornecidos, se r_1 for maior do que a probabilidade de cruzamento, caso contrário ela retorna o cromossomo formado pela inserção da lista $[y_{j_2}, ..., y_{j_3}]$ na posição j_1 da lista $[x]$.

As operações de seleção e substituição adotadas pelo algoritmo genético baseado em aprendizagem por *linkage* pode ser qualquer uma das existentes na literatura. Uma extensão deste algoritmo é o algoritmo genético baseado em aprendizagem por *linkage* com compressão de *introns* proposto por Lobo et al [109, 142] que trabalha com uma lista de pares (j, x) e números inteiros, na qual os pedaços de *introns* existentes na lista que representa o cromossomo do algoritmo genético baseado em aprendizagem por *linkage* são substituídos por um número inteiro correspondendo ao tamanho do referido pedaço de *introns*.

2.4 Algoritmos Genéticos Eficientes

2.4.1 Algoritmo Genético Paralelo

A proposta de uma implementação paralela do algoritmo genético, denominada de algoritmo genético paralelo, apresenta-se como uma resposta à necessidade de manter a diversidade na população, de diminuir o tempo de execução e de aumentar a qualidade dos resultados encontrados pelo algoritmo genético [23]. Existem muitos modelos de algoritmo genético paralelo, nesta seção serão discutidos somente os modelos

mestre escravo e *subpopulação estática*.

O modelo *mestre escravo* [12, 77, 22], também conhecido como algoritmo genético distribuído, paraleliza o cálculo da adaptação dos cromossomos e/ou a ação dos operadores genéticos entre os processadores *escravos*, ficando o processador *mestre* responsável pelo controle das atividades realizadas pelos processadores *escravos*, bem como pela execução serial das funções de seleção e substituição.

Quando o processador *mestre* espera que os processadores *escravos* terminem a sua tarefa para construir a próxima população, então se diz que este modelo é síncrono, senão diz-se que ele é assíncrono. Para otimizar o atraso da comunicação entre o processador *mestre* e os processadores *escravos* Cantú-Paz[24] sugere que o número ideal de processadores para um modelo *mestre escravo* é $\Upsilon = \sqrt{\frac{n \times Time_r}{Time_c}}$, onde n é o tamanho da população, $Time_r$ é o tempo requerido para o processador executar a função paralelizada de maior custo computacional e $Time_c$ é o tempo requerido para a comunicação entre os processadores.

O modelo de *subpopulação estática* caracteriza-se por dividir a população em um determinado número de subpopulações (*demes*), as quais evoluem de maneira independente. A competição entre os cromossomos de subpopulações diferente ocorre pela migração de cromossomos entre todas as subpopulações, ou somente entre as subpopulações vizinhas, ou pela existência de uma subpopulação cujos cromossomos participam de todas as outras populações. A migração de cromossomos é controlada pela definição dos seguintes parâmetros: topologia de conexão entre as subpopulações [114, 33], taxa de migração [24, 27], estratégia de migração (cromossomo mais adaptado, escolha aleatória, etc.) [26, 25], e freqüência de migração [164, 129, 33].

2.4.2 Algoritmo Genético de Tempo Contínuo

Os algoritmos genéticos de tempo contínuo agrupam as populações trabalhadas desde a sua inicialização até a sua convergência em épocas [161]. As épocas são intervalos evolucionários dentro do processo de evolução da população em que a diversidade da população é perdida. Quando uma época é concluída os algoritmos genéticos de tempo contínuo aplicam os operadores genéticos de continuidade, que são operações de mutação com probabilidade aproximadamente igual a 100%, responsáveis por perturbar as informações codificadas nos cromossomos da população para gerar a próxima população a ser manipulada pelo algoritmo genético.

O que há por trás da idéia de época é que a população construída na iteração anterior do algoritmo genético convergiu prematuramente para um resultado falso e precisa ser perturbada para permitir que os operadores

genéticos de cruzamento e/ou mutação possam dirigir-se para o resultado ótimo do espaço de busca investigado [7]. As medidas usadas para determinar o fim de uma época são a diferença entre a adaptação média da população e o valor da maior adaptação existente na população, o valor da adaptação do cromossomo menos adaptado e mais adaptado da população atual, etc. Quando o valor de uma destas medidas é inferior ao valor estabelecido durante a concepção ou a execução do algoritmo genético de tempo contínuo, então a época vigente é encerrada e uma nova época é iniciada.

$t \leftarrow 0$
gera p
calcule o valor da adaptação dos cromossomos de p
enquanto (condição de parada não for satisfeita) **faça**
    enquanto (uma época não tiver sido terminada) **faça**
        $\check{p} \leftarrow$ seleciona h cromossomos da população p
        $\dot{p} \leftarrow$ aplica a operação de cruzamento e mutação sobre os cromossomos da população $\check{p}$
        $p \leftarrow$ substitui l cromossomos da população p por l cromossomos da população $\dot{p}$
        calcule o valor da adaptação dos cromossomos de p
    fim do enquanto
    $p \leftarrow$ aplica a operação de continuidade sobre os cromossomos da população p
    $t \leftarrow t + 1$
fim do enquanto

Figura 2.4: Algoritmo genético de tempo contínuo

2.4.3 *Hardware* Evolutivo

O termo *hardware* evolutivo é usado para denominar tanto a aplicação de algoritmos de computação evolutiva como mecanismo de controle da configuração de FPGA (ver Apêndice B), denominado de *hardware* evolutivo de granularidade fina, como o desenvolvimento de um *hardware* para implementar um algoritmo genético específico, chamado de *hardware* evolutivo de granularidade grossa. Esta seção irá apresentar algumas arquiteturas de *hardware* evolutivo de granularidade grossa encontradas na literatura.

Graham [67] propõe uma implementação de um algoritmo genético em hardware para resolver problemas combinatoriais, composta de 4 blocos de memória e 4 FPGAs (*Field Programmable Gate Array*), como mostra

a Figura 2.5.

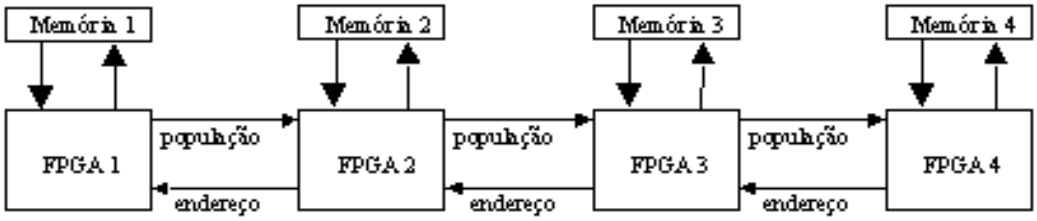


Figura 2.5: Arquitetura de Graham

Nesta arquitetura, inicialmente todos os blocos de memória contêm uma cópia da população inicial. A FPGA 1 executa a rotina de seleção de Monte Carlo (Definição 2.2.15), o endereço dos cromossomos selecionados são enviados para a FPGA 2. A FPGA 2 realiza a operação de cruzamento ordenado (Definição 2.2.25). Os cromossomos resultantes da operação de cruzamento são passados para a FPGA 3. A FPGA 3 aplica a operação de mutação por inversão (Definição 2.2.29) sobre os cromossomos enviados pela FPGA 1 e FPGA 2. Em seguida, ela calcula a adaptação de todos os cromossomos armazenados em sua memória e os envia para a FPGA 4 junto com o valor de sua adaptação. A FPGA 4 armazena os cromossomos gerados junto com seu valor de adaptação em um local da memória reservado para a população auxiliar até que uma geração tenha sido completada. Neste momento é iniciada a rotina de substituição elitista (Definição 2.2.11). Se uma solução aceitável para o problema não tiver sido encontrada, ou se o número máximo de iterações estipulado não tiver sido atingido, a nova população é enviada para a memória das FPGAs 1, 2 e 3, e o processo se repete.

Em Rintala [134] encontra-se uma outra arquitetura *pipeline* para um algoritmo genético (Figura 2.6) composta de 4 bancos de registro, 4 FPGA's, uma unidade de controle, um gerador de números aleatórios e uma memória. Nesta arquitetura a memória utilizada é uma RAM síncrona com barramentos de I/O separados e o gerador de números aleatórios utilizados é um linear *shift-register* (LSHR) com três registradores, para garantir uma periodicidade longa.

Inicialmente a população é carregada na memória, em seguida dois cromossomos são selecionados aleatoriamente e armazenados no segundo banco de registro (da esquerda para a direita). Depois que os cromossomos selecionados são armazenados, a unidade de controle envia um sinal para o módulo cruzamento e mutação. Assim que o módulo cruzamento e mutação recebe o sinal da unidade de controle, ele aplica a operação de cruzamento de um ponto de corte (Definição 2.2.18) sobre os cromossomos selecionados e algumas vezes a operação de mutação por complemento (Definição 2.2.30) sobre os cromossomos resultantes do cruzamento. Em

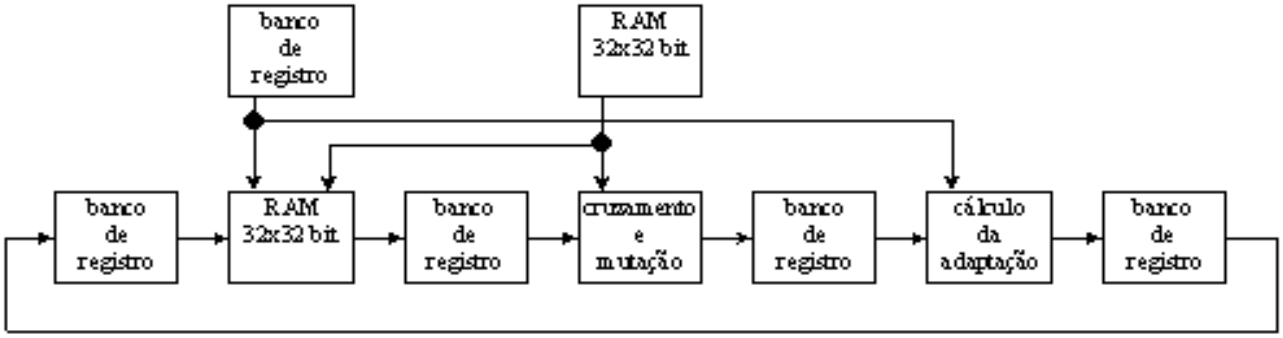


Figura 2.6: Arquitetura de Rintala

seguida o módulo cruzamento e mutação armazena os dois cromossomos gerados pelos operadores genéticos e os dois cromossomos selecionados para o terceiro banco de registro.

Após o armazenamento dos cromossomos gerados e selecionados no terceiro banco de registro, a unidade de controle envia um sinal para o módulo de cálculo de adaptação, onde a adaptação destes cromossomos é calculada. Em seguida, aplica-se a operação de substituição determinística (Definição 2.2.34) sobre estes cromossomos e armazena-se os cromossomos resultantes no quarto banco de registro. Estes cromossomos resultantes serão passados para o primeiro banco de registro e escritos nas mesmas posições da memória ocupadas pelos cromossomos selecionados. Este processo se repete até que uma solução satisfatória seja encontrada.

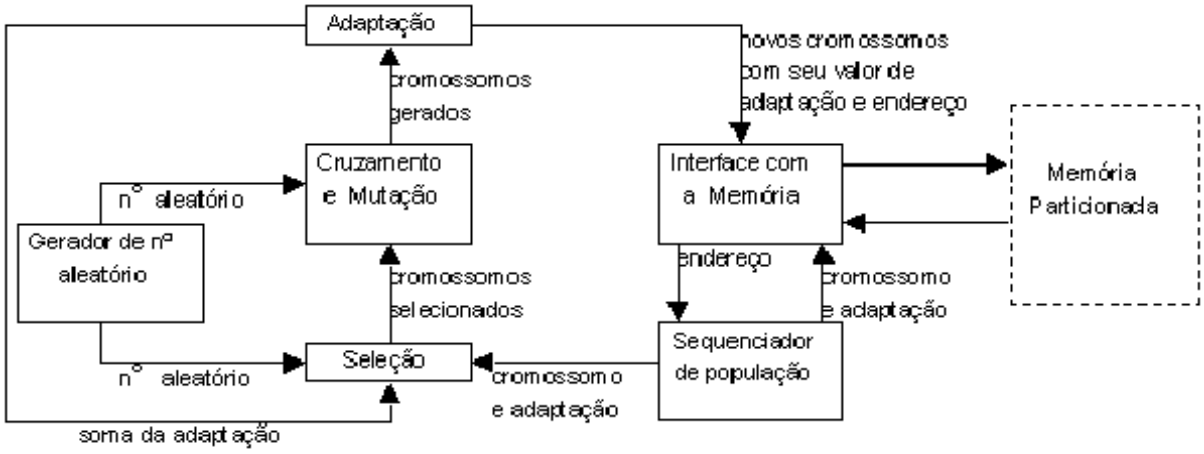


Figura 2.7: Arquitetura de Scott

Scott [147] sugere um sistema (Figura 2.7) que permite executar em hardware o modelo de um algoritmo genético simples [60] para qualquer problema, composto de uma interface para interagir com o usuário recebendo a especificação do problema e exibindo o resultado encontrado; e de um tradutor para traduzir a

especificação do usuário em uma especificação implementável no *hardware* como uma descrição em VHDL, a qual será carregada em uma placa para posterior emulação. Durante a emulação serão acessados endereços da memória reservados para os parâmetros fornecidos pelo usuário e os parâmetros gerados pelo sistema.

O algoritmo genético simples trabalha com um cromossomo representado por um vetor binário (Definição 2.2.1); uma população representada por um vetor (Definição 2.2.6); com o mecanismo de seleção de Monte Carlo (Definição 2.2.15); com o cruzamento de um ponto de corte (Definição 2.2.18), com a mutação por complemento (Definição 2.2.30); e com uma operação de substituição que simplesmente copia os cromossomos descendentes para o endereço da memória ocupado pelos cromossomos ancestrais.

Capítulo 3

Um Algoritmo Genético Baseado em Tipos Abstratos de Dados

3.1 Introdução

Com os modelos de algoritmos genéticos descritos no capítulo anterior em mente, pode-se concluir que estes algoritmos estão sedimentados sobre os seguintes princípios:

- para todo problema existe uma função de codificação que mapeia os possíveis resultados do problema na representação adotada para o cromossomo,
- para todo cromossomo existe uma função de adaptação, a qual informa quão adaptado ao ambiente este cromossomo está,
- os cromossomos mais adaptados geram descendentes (pela ação dos operadores genéticos) os quais tendem a herdar as características adaptadas do cromossomo ancestral, e
- os cromossomos menos adaptados tendem a desaparecer.

O primeiro princípio está relacionado ao debate existente hoje na comunidade de algoritmos evolucionários sobre qual é a melhor representação a ser adotada para o cromossomo, se um vetor de comprimento fixo ou variado com um alfabeto binário ou não-binário. O principal argumento a favor da representação binária defendida por Goldberg [57] é que a quantidade de *schemata* avaliados pelo algoritmo genético em cada iteração quando o cromossomo for binário é muito maior do que a quantidade de *schemata* avaliados quando a representação adotada para o cromossomo for não-binária. Por exemplo, para representar uma variável com quatro estados o número de *schemata* avaliados em uma representação não-binária seria cinco (1,2,3,4 e ♯), onde ♯ é

o símbolo *don't care* que pode ser substituído por qualquer símbolo do alfabeto adotado para o cromossomo, ao passo que o número de *schemata* avaliados em uma representação binária para esta mesma variável seria nove (00,01,10,11,♯0,0♯,♯1,1♯ e ♯♯). Este argumento foi invalidado por Antonisse [4], que propôs uma nova interpretação para o símbolo ♯, a qual associava o símbolo ♯ a um subconjunto de cardinalidade maior do que dois e menor ou igual à cardinalidade do alfabeto adotado. Para diferenciar os distintos alfabetos relacionados a cada símbolo ♯ do *schema* de um cromossomo, Antonisse propõe que cada símbolo ♯ em um *schema* seja indexado pelo seu alfabeto. Segundo esta visão, a quantidade de *schemata* avaliados para um alfabeto não-binário é superior à quantidade avaliada para um binário. Por exemplo, para o alfabeto quaternário do exemplo acima o número de *schemata* avaliados passaria a ser quinze (1,2,3,4,$\sharp_{\{1,2\}}$,$\sharp_{\{1,3\}}$,$\sharp_{\{1,4\}}$, $\sharp_{\{2,3\}}$,$\sharp_{\{2,4\}}$,$\sharp_{\{3,4\}}$, $\sharp_{\{1,2,3\}}$,$\sharp_{\{1,2,4\}}$, $\sharp_{\{1,3,4\}}$,$\sharp_{\{2,3,4\}}$ e $\sharp_{\{1,2,3,4\}}$), enquanto o número de *schemata* avaliados para o alfabeto binário continuaria sendo nove (00,01,10,11,$\sharp_{\{1,2\}}$0, 0$\sharp_{\{1,2\}}$,$\sharp_{\{1,2\}}$1,1$\sharp_{\{1,2\}}$ e $\sharp_{\{1,2\}}\sharp_{\{1,2\}}$).

A posição adotada neste trabalho com respeito a esta questão é que o argumento a respeito do número de *schemata* avaliados não parece suficiente para balizar uma decisão sobre qual representação deve ser adotada para o cromossomo, já que dentre estes *schemata* podem existir algumas informações incorretas para a busca a ser realizada pelo algoritmo genético, resultando em uma convergência para um falso ótimo. A experiência indica que para cada classe de problema, dependendo da solução adotada, uma dada representação parece ser mais adequada do que a outra [137, 138]. Assim, ao invés de propor-se um novo modelo de algoritmo genético comprometido com uma determinada representação, preferiu-se simplesmente adotar o conceito de tipo abstrato de dados e uma representação do cromossomo estratificada em três níveis de percepção (cromossomo, gene e base), para modelar uma representação do cromossomo independente da solução adotada e do problema a ser tratado. Dependendo da interpretação dada aos tipos (cromossomo, gene e base) é possível reproduzir qualquer uma das representações de cromossomos existentes.

O segundo princípio estabelece que a função de adaptação de um cromossomo deve ser compatível com o ambiente no qual este está inserido; isto sugere que se o ambiente mudar, tal função deveria em princípio ser capaz de redirecionar a sua busca à procura do cromossomo mais adaptado para a nova configuração do ambiente. Um exemplo típico desse tipo de problema seria construir um algoritmo genético para monitorar os sinais vitais de um paciente na unidade de tratamento intensivo (UTI) de um hospital. Note que dependendo da evolução do quadro clínico do paciente, os sinais monitorados podem evoluir para o diagnóstico tanto de uma parada cardíaca como de uma deficiência respiratória. Mas como considerar tais alterações ambientais, se o

ambiente não é considerado pelo algoritmo genético?

O modelo de algoritmo genético proposto aqui trabalha sobre um ambiente definido como uma estrutura na qual um dos componentes é a população. Segundo este modelo, as mudanças ambientais são vistas como o marco do início de um novo período de evolução durante o qual os cromossomos da população atual irão sofrer a ação dos operadores genéticos com o intuito de construir uma nova população formada somente por cromossomos que satisfazem aos requisitos do ambiente atual. Após o período de evolução vem o período de estagnação, durante o qual a população não evolui. O período de estagnação é finalizado quando uma nova alteração ambiental ocorre, dando início a um novo ciclo de um período de evolução seguido por um período de estagnação.

O resultado do problema para o ambiente atual é o cromossomo mais adaptado na população de estagnação atingida, se esta for compatível com uma condição de parada, a qual deverá ser definida durante o período de concepção do GAADT. A presença desta condição de parada se faz necessária porque o intuito aqui não é imitar a natureza, que tem toda a eternidade para progredir, mas sim resolver problemas.

A cada período de evolução do algoritmo existe uma população de cromossomos mortos associada. Os cromossomos desta população são provenientes de uma das populações já trabalhadas pelo algoritmo genético, os quais foram avaliadas e descartadas devido ao seu baixo grau de adaptação às configurações atuais do ambiente. A presença da população de cromossomos mortos é justificada pela necessidade de evitar que seus cromossomos possam reaparecer nas próximas populações trabalhadas pelo algoritmo genético.

O terceiro princípio fala sobre a hereditariedade das características adaptadas ao ambiente dos cromossomos pais que devem ser passadas para os cromossomos filhos. Sobre este ponto, é importante ressaltar que os operadores de cruzamento encontrados na literatura só se preocupam com o fato de que as características apresentadas nos cromossomos filhos estejam presentes nos cromossomos pais, sem se preocupar com quanto a referida característica contribui para a adaptação do cromossomo-pai ao ambiente.

O operador de cruzamento apresentado neste trabalho constrói novos cromossomos somente com as características responsáveis pela adaptação dos cromossomos pais ao ambiente, as quais serão denominadas de genes dominantes. Uma função denominada grau de adaptação do gene é definida para informar quanto uma dada característica pode contribuir com a adaptação do cromossomo ao ambiente.

O quarto princípio descarta a possibilidade dos cromossomos não adaptados ao ambiente evoluírem e até conduzirem à geração de cromossomos mais adaptados do que os cromossomos mais adaptados da população

atual. Neste trabalho os cromossomos não adaptados antes de desaparecerem serão todos submetidos à ação do operador genético de mutação, como uma forma de garantir a presença das características adaptadas ao ambiente destes organismos nas próximas gerações.

O algoritmo genético aqui proposto irá registrar a história genealógica (táxon) dos cromossomos da população atual, para que uma explicação sobre o resultado encontrado para um dado problema possa ser gerada sempre que necessário. Um exemplo de problema onde a explicação do resultado encontrado é relevante seria construir um algoritmo genético para exibir os erros dos exercícios de soma de dois termos ($a_4a_3a_2a_1a_0$ e $b_4b_3b_2b_1b_0$) propostos por um sistema tutor a uma criança. O papel do algoritmo genético seria operar os termos da soma de modo a convergir para o resultado declarado pela criança ($c_5c_4c_3c_2c_1c_0$). Ao atingir a população de estagnação, o sistema tutor não estaria interessado nos cromossomos obtidos pelo algoritmo, mas sim na história genealógica destes cromossomos, que poderia dizer entre outras coisas, que a soma $a_2+b_2 = c_2$ está errada porque o elemento *vai-um* de $a_1 + b_1$ não foi considerado, ou porque a soma $a_2 + b_2 = c_2$ está incorreta, etc.

A Seção 3.3 deste capítulo descreve os tipos base, gene, cromossomo e população adotada pelo GAADT. A Seção 3.4 define os operadores genéticos que trabalham sobre os tipos definidos na Seção 3.3. A Seção 3.5 apresenta a estrutura do ambiente trabalhado pelo algoritmo genético proposto e a Seção 3.6 a definição do algoritmo propriamente dito. Em todas estas seções será usado como exemplo ilustrativo o problema do caixeiro viajante [70] enunciado na Seção 3.2 e a linguagem Z [153], como formalismo para a especificação dos estados e das operações do GAADT de suas instanciações.

3.2 O Problema do Caixeiro Viajante

O problema do caixeiro viajante pode ser resumido da seguinte forma: Dados um conjunto N de cidades e uma matriz de distâncias formada pelas ligações de cada par de cidades deste conjunto, construa a menor rota para o caixeiro percorrer as cidades de N, que parta de uma cidade n ($n \in N$), que passe por todas as outras cidades uma única vez e que termine na cidade n de origem.

Para um conjunto N de cardinalidade φ, a rota a ser percorrida pelo caixeiro seria composta por $\varphi+1$ pontos de passagem. O primeiro e o último ponto de passagem seriam preenchidos com o mesmo nome de cidade, neste caso n. Isto, por ser uma ligação comum a qualquer roteiro, não influencia no cálculo do número de rotas

possíveis. O segundo ponto de passagem da rota pode ser preenchido com qualquer uma das cidades ainda não visitadas, correspondendo a $\varphi - 1$ alternativas. O terceiro ponto de passagem da rota pode ser preenchido com qualquer uma das cidades ainda não visitadas, correspondendo a $\varphi - 2$ alternativa. Prosseguindo com este raciocínio, tem-se que, no momento do preenchimento do φ-ésimo ponto de passagem da rota, só existirá uma única cidade ainda não visitada para ocupá-lo. Pode-se concluir, então, que o número de rotas analisadas para um conjunto N de cardinalidade φ é $(\varphi - 1) \times (\varphi - 2) \times \ldots \times 1 = (\varphi - 1)!$.

A importância atribuída ao problema do caixeiro viajante deve-se ao fato deste problema servir como um ícone para os problemas pertencentes à categoria dos problemas NP-completos [94]. Isto porque o número de rotas que devem ser analisadas para resolver este problema cresce a uma proporção fatorial em função da cardinalidade do conjunto N de cidades.

3.3 Tipos Básicos

Neste trabalho, os cromossomos serão representados por seu material genético, o qual têm nas bases suas unidades elementares de formação.

Definição 3.3.1. (Base) - Uma base B é o conjunto de todas as unidades genéticas elementares que podem ser usadas na formação do material genético dos cromossomos de uma população.

A base será especificada por um *given set*, $[B]$, uma vez que este tipo básico da linguagem Z nos permite trabalhar com o tipo base sem que se precise dizer a natureza ou as propriedades dos objetos que ele contém.

Por exemplo, no caso do problema do caixeiro viajante, envolvendo quatro cidades ($CidadeA$, $CidadeB$, $CidadeC$, $CidadeD$), o tipo base pode ser instanciado como o conjunto composto pelos nomes destas cidades, $B ::= CidadeA \mid CidadeB \mid CidadeC \mid CidadeD$.

Os elementos de uma base se agrupam em seqüências para formar as características (genes) dos cromossomos. Mas nem toda seqüência de bases representa uma característica para o cromossomo. Portanto, deve existir uma *lei de formação* para indicar como as bases devem ser agrupadas para formar uma dada característica. Neste trabalho, a *lei de formação* de características será representada pelo conjunto de *Axiomas de Formação de Genes* (AFG), o qual deverá ser definido para cada caso de acordo com a semântica atribuída ao gene. A assinatura da definição axiomática do conjunto de axiomas de formação de genes é $AFG : \mathbb{P}(\mathsf{seq}\ B)$.

Note que as propriedades desta definição só podem ser completadas no momento da instanciação do algoritmo genético a um problema específico. Por exemplo, no caso do caixeiro viajante o AFG deveria estabelecer que as seqüências válidas são aquelas com tamanho igual a 2, tal que a base da primeira posição desta seqüência deve ser diferente da base que ocupa a segunda posição. A semântica associada a cada seqüência $\langle b_1, b_2 \rangle$ assim descrita é que existe um caminho da cidade b_1 para a cidade b_2.

$$\begin{array}{|l} AFG : \mathbb{P}(\ \mathsf{seq}\ B) \\ \hline \forall g : AFG \bullet \#g = 2 \end{array}$$

Definição 3.3.2. (Gene) - Um gene g é uma seqüência formada pelos elementos da base que pertence ao conjunto AFG.

Assim, para que as operações do GAADT possam manipular os objetos pertencentes ao tipo gene, é preciso primeiro defini-lo, o que é feito pela seguinte definição abreviada: $G == AFG$. Como pode ser observado o tipo gene irá variar dependendo das restrições impostas pelo conjunto AFG aos seus elementos.

Os genes são agrupados em conjuntos para formar os cromossomos da população. O conjunto de genes $\{g_1, g_2, \ldots, g_n\}$ que compõe um dado cromossomo serve para identificar este cromossomo dentro da população. A identidade dos cromossomos será usada para impedir que várias cópias de um cromossomo possam coexistir ou renascer na população em qualquer tempo durante o processo de evolução da mesma na busca por um cromossomo mais adaptado.

As características apresentadas pelos cromossomos de uma população servem também para classificá-los em grupos taxonômicos (espécies e famílias) em função do grau de similaridade das características compartilhadas pelos mesmos. Este fato conduziu alguns pesquisadores a concluir erroneamente que todos os cromossomos com alto grau de similaridade genética (e morfológica) pertenceriam ao mesmo grupo taxonômico [66]. O contra exemplo mais forte a esta conclusão é o alto grau de similaridade existente entre o homem e o chimpanzé, os quais não pertencem ao mesmo grupo taxonômico.

Após muitas tentativas, mal sucedidas, de diferenciar estas espécies pela característica *inteligência* atribuída ao homem, King e Wilson [98] concluíram que a principal diferença entre o homem e o chimpanzé não está

em suas características genéticas e morfológicas, mas sim no seu desenvolvimento embrionário. Segundo King e Wilson, existe um *relógio biológico* que ativa o início e fim da formação de uma característica em cada uma destas espécies em tempos embrionários diferentes. Tal *relógio biológico* deve também garantir que o cruzamento de dois cromossomos de uma mesma espécie resulte em um cromossomo da mesma espécie.

Neste trabalho, o *relógio biológico* do cromossomo será representado pelo conjunto de *Axiomas de Formação de Cromossomos* (AFC), o qual deverá ser definido para cada situação de acordo com a semântica adotada para o cromossomo.

Assim como o AFG, o AFC também será descrito por uma definição axiomática cujas propriedades só serão incluídas no momento da sua instanciação, apresentando o seguinte traço $AFC : \mathbb{P}(\mathbb{P}(G))$.

Por exemplo, um cromossomo para o problema do caixeiro viajante é qualquer conjunto de genes, tal que para todo elemento b do tipo base o número de ocorrências de b como cidade de partida e o número de ocorrências de b como cidade de chegada são ambos iguais a 1.

$$
\begin{array}{|l}
AFC : \mathbb{P}(\mathbb{P}(G)) \\
\hline
\forall c : AFC;\ b : B \bullet \\
\#\{g : c \mid \exists a : B \bullet g = \langle b, a \rangle\} = 1 \land \#\{g : c \mid \exists a : B \bullet g = \langle a, b \rangle\} = 1
\end{array}
$$

Definição 3.3.3. (Cromossomo) - Um cromossomo c é um conjunto de genes que obedece às condições estabelecidas pelo *AFC*.

O tipo cromossomo é definido como o conjunto de todos os conjuntos formados por objetos do tipo abstrato gene que pertencem ao conjunto AFC, $C == AFC$.

Os cromossomos são agrupados em conjuntos para formar uma população. Esta representação para a população irá garantir a imparcialidade na avaliação dos cromossomos que compõem a população, já que cada cromossomo só poderá ocorrer uma vez na população.

Definição 3.3.4. (População) - Uma população P_j é um conjunto de cromossomos construídos conforme descrito na Definição 3.3.3

O tipo população é o conjunto formado por todos os conjuntos formados por objetos do tipo cromossomo (ou seja, $P == \mathbb{P}(\mathbb{P}(C))$), que são possíveis resultados para o problema em foco segundo a interpretação adotada para os tipos C, G e B. Deve-se ressaltar que a geração da população vazia pelo algoritmo indica que a interpretação adotada para o problema está errada.

3.4 Operadores Genéticos

O GAADT trabalha com dois tipos de operadores genéticos: o de reprodução e o de mutação. O operador genético de reprodução caracteriza-se por combinar os genes de dois cromossomos (cromossomos-pai) para formar outros cromossomos (cromossomos-filho), enquanto que o operador genético de mutação caracteriza-se por alterar a identidade de um cromossomo para formar um outro cromossomo (cromossomo-mutante).

O gene dos cromossomos-pai para uma dada característica que fará parte dos cromossomos-filho é aquele que melhor satisfaz as restrições do problema sobre a característica expressa por este gene, o qual será denominado de gene-dominante. Este gene não apresenta o mesmo nível epistemológico do gene dominante proposto por Mendel em seu trabalho com ervilhas [10], já que existe uma grande diferença entre dizer que um gene g_1 satisfaz melhor as restrições do problema do que um gene g_2 e dizer que o fator hereditário de um gene g_1 é superior ao fator hereditário de um gene g_2.

Dados dois genes g_1 e g_2, que expressem uma mesma característica com diferentes fenótipos, diz-se que um gene g_1 melhor satisfaz as restrições do problema do que o gene g_2, se o grau de adaptação do gene g_1 for superior ou igual ao grau de adaptação do gene g_2.

Neste trabalho, o grau de adaptação de um gene é dado por uma função *grau* e será considerada a existência de um gene g_λ que será usado para representar um gene que não expressa qualquer característica, de forma que a sua presença ou ausência não altera a identidade do cromossomo, o qual satisfaz as restrições impostas pelo conjunto de axiomas de formação de genes. Seu grau de adaptação é menor que o grau de adaptação de qualquer outro elemento do tipo gene. Tal gene será denominado de gene-inócuo.

Definição 3.4.1. (Grau) - O grau de adaptação de um gene é uma função *grau* do seguinte tipo:
$grau : G \rightarrow K$ tal que, a cada gene g, $g \in G$, é associado um único número k, $k \in K$ (K é um corpo ordenado[1]), chamado de $grau(g)$ e que reflete, segundo a interpretação adotada para o problema, uma estratificação

[1]Corpo ordenado - é uma estrutura algébrica, com duas operações, sem divisores próprios de zero e munido de uma ordem. Ex:

comparativa entre a adaptação dos genes.

O gene inócuo será uma constante do sistema, cujo valor deverá ser definido no momento da instanciação do algoritmo a um dado problema. Por enquanto, basta saber que ele pertence ao tipo gene, que tem grau de adaptação igual ao elemento neutro do corpo K para a operação de adição, e que para todo gene g com $g \neq g_\lambda$ $grau(g)$ é maior do que $grau(g_\lambda)$.

Por exemplo, para o problema do caixeiro viajante será preciso redefinir o tipo base e o AFG, para que o gene inócuo possa ser definido. O tipo base é acrescido do elemento que irá formar a seqüência do gene inócuo, $B ::= CidadeA \mid CidadeB \mid CidadeC \mid CidadeD \mid CidadeInocua$.

O conjunto de axiomas de formação de gene irá conter mais um predicado, para informar a composição do gene inócuo, que neste caso é a seqüência $\langle CidadeInocua, CidadeInocua \rangle$.

$$
\begin{array}{|l}
AFG : \mathbb{P}(\ \mathsf{seq}\ B) \\
\hline
(\forall g : AFG \mid CidadeInocua \in\ \mathsf{ran}\ g \bullet \#\ \mathsf{ran}\ g = 1) \land \\
(\forall g : AFG \mid CidadeInocua \notin\ \mathsf{ran}\ g \bullet \#\ \mathsf{ran}\ g = 2\})
\end{array}
$$

E o gene inócuo é definido através da constante g_λ:

$$
\begin{array}{|l}
g_\lambda : G \\
\hline
[geneInocuoDef] \\
g_\lambda = \langle CidadeInocua, CidadeInocua \rangle
\end{array}
$$

Nesta especificação em Z, o corpo K considerado é o conjunto dos números racionais. Mas como este conjunto não faz parte do conjunto de números fornecidos pela linguagem Z, então se construiu este conjunto a partir do conjunto dos números inteiros pela definição abreviada $\mathbb{Q}$, $\mathbb{Q} == \{x, y : \mathbb{Z} \mid y > 0\}$. E para

$\langle \mathbb{R}, \leq, +, \times, 0, 1 \rangle$. Maiores detalhes ver em [136, 167]

manipular as variáveis do tipo $\mathbb{Q}$ são definidas as operações e relações abaixo.

$$
\begin{array}{|l}
adicao\mathbb{Q} : \mathbb{Q} \times \mathbb{Q} \pfun \mathbb{Q} \\
subtracao\mathbb{Q} : \mathbb{Q} \times \mathbb{Q} \pfun \mathbb{Q} \\
multiplicacao\mathbb{Q} : \mathbb{Q} \times \mathbb{Q} \pfun \mathbb{Q} \\
divisao\mathbb{Q} : \mathbb{Q} \times \mathbb{Q} \pfun \mathbb{Q} \\
mesmo\mathbb{Q} : \mathbb{Q} \rel \mathbb{Q} \\
maior\mathbb{Q} : \mathbb{Q} \rel \mathbb{Q} \\
menor\mathbb{Q} : \mathbb{Q} \rel \mathbb{Q} \\
\hline
[AdicaoRacionalDef] \\
\forall x, y : \mathbb{Q} \bullet \\
adicao\mathbb{Q}(x, y) = (first(x) * second(y) + first(y) * second(x), second(x) * second(y)) \\
[SubtracaoRacionalDef] \\
\forall x, y : \mathbb{Q} \bullet \\
subtracao\mathbb{Q}(x, y) = (first(x) * second(y) - first(y) * second(x), second(x) * second(y)) \\
[MultiplicacaoRacionalDef] \\
\forall x, y : \mathbb{Q} \bullet multiplicacao\mathbb{Q}(x, y) = (first(x) * first(y), second(x) * second(y)) \\
[DivisaoRacionalDef] \\
\forall x, y : \mathbb{Q} \bullet divisao\mathbb{Q}(x, y) = (first(x) * second(y), second(x) * first(y)) \\
[MesmoRacionalDef] \\
\forall x, y : \mathbb{Q} \mid first(x) * second(y) = first(y) * second(x) \bullet (x, y) \in mesmo\mathbb{Q} \\
[MaiorRacionalDef] \\
\forall x, y : \mathbb{Q} \mid first(x) * second(y) > first(y) * second(y) \bullet (x, y) \in maior\mathbb{Q} \\
[MenorRacionalDef] \\
\forall x, y : \mathbb{Q} \mid first(x) * second(y) < first(y) * second(y) \bullet (x, y) \in menor\mathbb{Q}
\end{array}
$$

Se o tipo $\mathbb{Q}$ é um corpo, então ele deve satisfazer as seguintes propriedades:

- A adição é associativa, isto é,

  $\forall x, y, z : \mathbb{Q} \bullet adicao\mathbb{Q}(adicao\mathbb{Q}(x, y), z) = adicao\mathbb{Q}(x, adicao\mathbb{Q}(y, z))$;

- Existe um elemento neutro $a_{neutro} = (0, 1)$ para a operação de adição, isto é,

  $\exists a_{neutro} : \mathbb{Q} \bullet \forall x : \mathbb{Q} \bullet adicao\mathbb{Q}(x, a_{neutro}) = adicao\mathbb{Q}(a_{neutro}, x) = x$;

- A adição tem inversa, isto é,

  $\forall x : \mathbb{Q} \bullet \exists y : \mathbb{Q} \bullet (adicao\mathbb{Q}(x, y), a_{neutro}) \in mesmo\mathbb{Q}$;

- A adição é unívoca, isto é,

  $\forall x, y : \mathbb{Q} \bullet adicao\mathbb{Q}(x, y) = adicao\mathbb{Q}(y, x)$;

- A multiplicação é associativa, isto é,

  $\forall x, y, z : \mathbb{Q} \bullet multiplicacao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), z) =$

  $multiplicacao\mathbb{Q}(x, multiplicacao\mathbb{Q}(y, z))$;

- A multiplicação é distributiva em relação à adição, isto é,

  $\forall x, y, z : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(x, adicao\mathbb{Q}(y, z)),$

  $adicao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), multiplicacao\mathbb{Q}(x, z))) \in mesmo\mathbb{Q}$

  e

  $\forall x, y, z : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(adicao\mathbb{Q}(y, z), x),$

  $adicao\mathbb{Q}(multiplicacao\mathbb{Q}(y, x), multiplicacao\mathbb{Q}(z, x))) \in mesmo\mathbb{Q}$;

- Existe um elemento neutro $m_{neutro} = (1, 1)$ para a operação de mutiplicação, isto é,

  $\exists m_{neutro} : \mathbb{Q} \bullet \forall x : \mathbb{Q} \bullet multiplicacao\mathbb{Q}(x, m_{neutro}) = multiplicacao\mathbb{Q}(m_{neutro}, x) = x$;

- A multiplicação tem inversa para todo elemento diferente dos pares cujo componente x é zero, isto é,

  $\forall x : \mathbb{Q} \mid first(x) \neq 0 \bullet \exists y : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(x, y), m_{neutro}) \in mesmo\mathbb{Q}$.

Note que na definição destas propriedades, a existência do elemento inverso, assim como a distributividade da operação de multiplicação em relação à operação de adição, foi enfraquecida pela presença da relação de $mesmo\mathbb{Q}$. A utilização da relação $mesmo\mathbb{Q}$ no lugar da relação $=$ deve-se ao fato de que os números racionais modelados aqui não são simplificados. A prova destas propriedades encontra-se no Apêndice C.

Com base nas operações definidas para o tipo $\mathbb{Q}$, define-se a função grau de adaptação de um gene como:

$$
\begin{array}{|l}
grau : G \pfun \mathbb{Q} \\
\hline
[grauInocuoProp1Def] \\
grau(g_\lambda) = (0, 1) \\
[grauInocuoProp2Def] \\
\forall g : G \bullet g \neq g_\lambda \implies (grau(g), grau(g_\lambda)) \in maior\mathbb{Q}
\end{array}
$$

Por exemplo, para o problema do caixeiro viajante, a função $grau$ deverá fornecer uma medida sobre a distância entra as cidades consideradas, de modo que a soma destas medidas informem a menor distância a ser percorrida pelo caixeiro. A distância entre as cidades $CidadeA$, $CidadeB$, $CidadeC$ e $CidadeD$ está contida na Tabela 3.1. Assim a função $grau(g = \langle b_1, b_2 \rangle) = \frac{1}{d(b_1,b_2)}$ para todo gene diferente do gene inócuo, e $grau(g_\lambda)$ é zero. Mas como a função grau foi especificada em Z como um mapeamento de gene em racional, então as frações $\frac{1}{d(b_1,b_2)}$ serão representadas em Z pelos números racionais $(1, d(b_1, b_2))$.

	$CidadeA$	$CidadeB$	$CidadeC$	$CidadeD$
$CidadeA$	0	3	7	5
$CidadeB$	3	0	2	1
$CidadeC$	7	2	0	4
$CidadeD$	5	1	4	0

Tabela 3.1: Distância entre as cidades

$$grau : G \pfun \mathbb{Q}$$

$$[GrauInocuoPro1Def]$$
$$grau(g_\lambda) = (0, 1)$$
$$[GrauABDef]$$
$$\forall g : G \mid \mathsf{ran}\ g = \{CidadeA, CidadeB\} \bullet grau(g) = (1, 3)$$
$$[GrauACDef]$$
$$\forall g : G \mid \mathsf{ran}\ g = \{CidadeA, CidadeC\} \bullet grau(g) = (1, 7)$$
$$[GrauADDef]$$
$$\forall g : G \mid \mathsf{ran}\ g = \{CidadeA, CidadeD\} \bullet grau(g) = (1, 5)$$
$$[GrauBCDef]$$
$$\forall g : G \mid \mathsf{ran}\ g = \{CidadeB, CidadeC\} \bullet grau(g) = (1, 2)$$
$$[GrauBDDef]$$
$$\forall g : G \mid \mathsf{ran}\ g = \{CidadeB, CidadeD\} \bullet grau(g) = (1, 1)$$
$$[GrauCDDef]$$
$$\forall g : G \mid \mathsf{ran}\ g = \{CidadeC, CidadeD\} \bullet grau(g) = (1, 4)$$

Lema 3.4.2. $\forall g : G \mid g \neq g_\lambda \bullet (grau(g), grau(g_\lambda)) \in maior\mathbb{Q}$

Por uma questão de nomenclatura, será também exibida a relação de equivalência cromossômica, denotada por $\equiv_C$, definida da seguinte maneira: $\equiv_C$: $C \leftrightarrow C$, tal que $c_1 \equiv_C c_2$, se e somente se, $c_1 - \{g_\lambda\}$=$c_2 - \{g_\lambda\}$. Esta relação será descrita por uma definição axiomática.

$$\equiv_C : C \leftrightarrow C$$

$$[eqDef]$$
$$\forall c_1, c_2 : C \bullet c_1 \setminus \{g_\lambda\} = c_2 \setminus \{g_\lambda\} \Leftrightarrow (c_1, c_2) \in \equiv_C$$

O gene dominante é identificado pela função $domi$ que recebe um par de genes, um de cada um dos cromossomos-pai, e retorna o gene de maior grau de adaptação se os genes fornecidos expressarem uma mesma

característica. Caso os genes fornecidos não expressem uma mesma característica, então a função $domi$ retornará g_λ.

Dados dois genes, diz-se que eles expressam uma mesma característica se existe um atributo relevante para o problema em foco que seja satisfeito pelos genes fornecidos. Neste trabalho, o atributo relevante para o problema é especificado pelo conjunto $atributoRelevante$, considerado uma constante do sistema, cujo conteúdo dependerá da interpretação adotada para o problema.

$$atributoRelevante : \mathbb{P}(\mathbb{P}\ G)$$

No caso do problema do caixeiro viajante, o conjunto $atributoRelevante$ exige que as cidades que compõem um dos genes analisados sejam as mesmas que compõem o outro gene, como ilustra a definição axiomática abaixo:

$$atributoRelevante : \mathbb{P}(\mathbb{P}\ G)$$

$$[atributoRelevanteDef]$$
$$\forall g_1, g_2 : G \bullet \mathsf{ran}\ g_1 = \mathsf{ran}\ g_2 \Leftrightarrow (\exists G_1 : atributoRelevante \bullet \{g_1, g_2\} \subseteq G_1)$$

A relação $mesma$ é especificada pela seguinte definição axiomática:

$$mesma : G \leftrightarrow G$$

$$[mesmaDef]$$
$$\forall g_1, g_2 : G \bullet (g_1, g_2) \in mesma \Leftrightarrow (\exists G_1 : atributoRelevante \bullet \{g_1, g_2\} \subseteq G_1)$$

Lema 3.4.3. $\forall g : G \bullet (g, g) \in mesma$.

Lema 3.4.4. $\forall g_1, g_2 : G \mid (g_1, g_2) \in mesma \bullet (g_2, g_1) \in mesma$.

Lema 3.4.5. $\forall g_1, g_2, g_3 : G \mid (g_1, g_2) \in mesma \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$.

Definição 3.4.6. (Dominante) - O gene dominante é uma função $domi$ do seguinte tipo:

$domi : G \times G \rightarrow G$

$$domi(g_1, g_2) = \begin{cases} g_\lambda & \text{se } (g_1, g_2) \notin mesma, \\ g_1 & \text{se } (g_1, g_2) \in mesma \wedge grau(g_1) \geq grau(g_2), \\ g_2 & \text{se } (g_1, g_2) \in mesma \wedge grau(g_1) < grau(g_2). \end{cases}$$

A especificação da função $domi$ é feita através de uma definição axiomática, na qual cada rótulo representa o predicado que descreve cada uma das alternativas do sistema que define esta função. Nesta especificação, a primeira alternativa foi decomposta em duas propriedades $domiIgualDef$ e $domiMaiorDef$ para facilitar a prova das propriedade da função $domi$ (mostradas no Apêndice C).

$$\begin{array}{|l} \\ domi : G \times G \nrightarrow G \\ \hline \\ [domiIgualDef] \\ \forall g_1, g_2 : G \mid (g_1, g_2) \in mesma \wedge (grau(g_1), grau(g_2)) \in mesmo\mathbb{Q}\bullet \\ domi(g_1, g_2) = g_1 \\ [domiMaiorDef] \\ \forall g_1, g_2 : G \mid (g_1, g_2) \in mesma \wedge (grau(g_1), grau(g_2)) \in maior\mathbb{Q}\bullet \\ domi(g_1, g_2) = g_1 \\ [domiMenorDef] \\ \forall g_1, g_2 : G \mid (g_1, g_2) \in mesma \wedge (grau(g_1), grau(g_2)) \in menor\mathbb{Q}\bullet \\ domi(g_1, g_2) = g_2 \\ [domiIncomparavelDef] \\ \forall g_1, g_2 : G \mid (g_1, g_2) \notin mesma \bullet domi(g_1, g_2) = g_\lambda \end{array}$$

Lema 3.4.7. $\forall g : G \mid (g, g) \in mesma \bullet domi(g, g) = g.$

Lema 3.4.8. $\forall g_1, g_2, g_3 : G \mid (g_1, g_2) \in mesma \wedge (g_2, g_3) \in mesma\bullet$

$domi(g_1, domi(g_2, g_3)) = domi(domi(g_1, g_2), g_3)$

A produção de novos cromossomos durante o processo evolutivo de uma população serve para direcionar a busca por cromossomos mais adaptados através da transmissão das características de maior grau de adaptação presentes nos cromossomos da população atual. A adaptação de um cromossomo é dada pela função *adapt*.

Definição 3.4.9. (Adaptação) - A adaptação de um cromossomo é uma função $adapt$ do seguinte tipo:

$adapt : C \rightarrow K$

$adapt(c) = \sum_{g \in c} \Theta_{c,g} \times grau(g)$

onde $\Theta_{c,g}$ é o peso com o qual o gene g contribui para a adaptação do cromossomo c.

Como nada se pode dizer sobre o valor do parâmetro peso usado no cálculo de adaptação de um cromossomo, antes de se saber a qual problema o GAADT será aplicado, já que este cálculo pode levar em consideração a presença ou ausência de um dado gene no cromossomo, então só se pode definir a assinatura da função Θ, como o mapeamento do par cromossomo e gene em um valor racional positivo. O conjunto dos números racionais positivos é especificado pela seguinte definição abreviada $\mathbb{Q}^+ = \{q : \mathbb{Q} \mid (q,(0,1)) \in maior\mathbb{Q} \vee (q,(0,1)) \in mesmo\mathbb{Q}\}$. Assim, a especificação desta função consistirá somente da sua assinatura, $\Theta : C \times G \nrightarrow \mathbb{Q}^+$.

Por exemplo, para o problema do caixeiro viajante o peso de qualquer gene no cromossomo será igual à (0,1), se este gene não pertencer ao cromossomo, caso contrário ele será (1,1).

$$\Theta : C \times G \nrightarrow \mathbb{Q}^+$$

$$[ThetaNaoPertenceDef]$$
$$\forall c : C;\ g : G \mid g \notin c \bullet \Theta(c,g) = (0,1)$$
$$[ThetaPertenceDef]$$
$$\forall c : C;\ g : G \mid g \in c \bullet \Theta(c,g) = (1,1)$$

A função de adaptação é definida pela composição das funções $adapt$ e $adaptRecursivo$. A função $adapt$ é usada simplesmente para preparar a entrada para a função $adaptRecursivo$, que calcula efetivamente o valor

da adaptação do cromossomo.

$$
\begin{array}{|l}
adaptRecursivo : C \times C \pfun \mathbb{Q}^+ \\
\hline
[adaptRecursivoFimDef] \\
\forall c : C \bullet adaptRecursivo(c, \emptyset) = (0, 1) \\
[adaptRecursivoDef] \\
\forall c_1, c_2 : C \bullet \exists g : c_2 \bullet \\
adaptRecursivo(c_1, c_2) = adicao\mathbb{Q}(adaptRecursivo(c_1, c_2 \setminus \{g\}), multiplicacao\mathbb{Q}(\Theta(c_1, g), grau(g)))
\end{array}
$$

$$
\begin{array}{|l}
adapt : C \pfun \mathbb{Q}^+ \\
\hline
[adaptVazioDef] \\
adapt(\emptyset) = (0, 1) \\
[adaptDef] \\
\forall c : C \bullet adapt(c) = adaptRecursivo(c, c)
\end{array}
$$

A operação de cruzamento recebe dois cromossomos-pai, aptos a cruzarem, e retorna uma população cujos cromossomos são formados somente pelos genes dominantes dos cromossomos fornecidos. Logo, para se definir esta função precisa-se antes definir uma função para selecionar os cromossomos aptos a cruzarem (seleção) e uma função para retornar o conjunto de genes dominantes para todas as características existentes nos cromossomos-pai (fecundação).

A função de seleção recebe uma população P_1 e retorna a subpopulação de P_1 formada pelos cromossomos que satisfazem um requisito do problema r, descrito por uma fórmula em lógica de primeira ordem, o qual indica quando um dado cromossomo é considerado apto a cruzar.

Definição 3.4.10. (Seleção) - A seleção dos cromossomos que satisfazem um predicado r é uma função sel do seguinte tipo:

$sel : \mathbb{P}(P) \times \mathbb{P}(P) \to \mathbb{P}(P)$

$sel(P_1, r) = P_1 \cap r.$

O requisito r será representado em Z pela seguinte definição axiomática: $r : \mathbb{P}(P)$. Por exemplo, o predicado que define o conjunto r para o problema do caixeiro viajante poderia ser: passar pela cidade $CidadeA$. Os cromossomos desta população serão então aqueles que têm entre seus genes pelo menos uma ocorrência da cidade $CidadeA$.

$$\begin{array}{|l} r : \mathbb{P}(P) \\ \hline \forall c : r \bullet \exists g : c \bullet CidadeA \in \mathsf{ran}\ g \end{array}$$

A operação de seleção é especificada pela função sel na definição axiomática abaixo:

$$\begin{array}{|l} sel : \mathbb{P}(P) \times \mathbb{P}(P) \pfun \mathbb{P}(P) \\ \hline [selDef] \\ \forall P_1 : \mathbb{P}(P) \bullet sel(P_1, r) = P_1 \cap r \end{array}$$

A função fecundação recebe dois cromossomos e retorna o conjunto de genes dominantes entre todos os genes dos cromossomos fornecidos.

Definição 3.4.11. (Fecundação) - A fecundação é uma função fec do seguinte tipo:

$fec : C \times C \rightarrow \mathbb{P}(G)$

$fec(c_1, c_2) = \{g \mid \forall g_1 \in c_1 \, \forall g_2 \in c_2 (g = domi(g_1, g_2))\}$

Na linguagem Z esta função é descrita pela seguinte definição axiomática:

$$\begin{array}{|l} fec : C \times C \pfun \mathbb{P}(G) \\ \hline [fecDef] \\ \forall c_1, c_2 : C \bullet fec(c_1, c_2) = \{g_1 : c_1;\ g_2 : c_2 \bullet domi(g_1, g_2)\} \end{array}$$

Lema 3.4.12. $\forall c : C \bullet (fec(c, c), c) \in \equiv_C$

Os cromossomos-pai aptos a cruzarem são representados pelo conjunto $MACHO$ e $FEMEA$, formados da seguinte forma: $MACHO = sel(P_1, M)$ e $FEMEA = sel(P_1, F)$, onde P_1 é uma subpopulação da população atual formada por cromossomos adaptados ao ambiente e, M e F são dois predicados sobre o tipo população pertencentes ao conjunto de requisitos do ambiente Rq, escritos em uma linguagem de primeira ordem.

$$\begin{array}{|l} \\ M : \mathbb{P}(P) \\ F : \mathbb{P}(P) \end{array}$$

Note que, dependendo da especificação dos requisitos do ambiente M e F, a reprodução gerada pode ser sexuada, assexuada ou mista. A reprodução sexuada ocorre quando $M \cap F = \emptyset$, a assexuada quando $M = F$, e a mista quando $M \cap F \neq \emptyset$ e $M \neq F$.

Por exemplo, para o problema do caixeiro viajante, o requisito M pode exigir que as cidades $CidadeA$ e $CidadeB$ façam parte da rota representada pelos seus cromossomos, enquanto F exige que as cidades $CidadeC$ e $CidadeD$ façam parte da rota representada pelos seus cromossomos. Assim, o cromossomo resultante deste cruzamento deve representar uma rota que passa pelas cidades $CidadeA$, $CidadeB$, $CidadeC$, e $CidadeD$.

$$\begin{array}{|l} \\ M : \mathbb{P}(P) \\ F : \mathbb{P}(P) \\ \hline \\ (\forall c : M \bullet \exists g_1, g_2 : c \bullet CidadeA \in \mathsf{ran}\ g_1 \wedge CidadeB \in \mathsf{ran}\ g_2) \wedge \\ (\forall c : F \bullet \exists g_1, g_2 : c \bullet CidadeC \in \mathsf{ran}\ g_1 \wedge CidadeD \in \mathsf{ran}\ g_2) \end{array}$$

Definição 3.4.13. (Cruzamento) - O cruzamento é uma função *cruz* do seguinte tipo:

$cruz : MACHO \times FEMEA \rightarrow P$

$cruz(c_1, c_2) = \{c \mid c \subseteq fec(c_1, c_2)\}$

A definição axiomática abaixo mostra a especificação da função $cruz$ em Z.

$$
\begin{array}{|l}
cruz : MACHO \times FEMEA \pfun P \\
\hline
[cruzDef] \\
\forall c_1 : MACHO;\ c_2 : FEMEA \bullet cruz(c_1, c_2) = \{c : C \mid c \subseteq fec(c_1, c_2)\}
\end{array}
$$

O operador genético de mutação, definido para o GAADT, é composto pelas funções de inserção, supressão e troca, tal que os cromossomos resultantes da ação destes operadores apresentarão parte dos genes contidos no cromossomo que lhe deu origem. A operação de inserção ins adiciona um conjunto de genes ao cromossomo de origem.

Definição 3.4.14. (Inserção) - A inserção é uma função ins do seguinte tipo:

$$
ins : C \times \mathbb{P}(G) \to C
$$

$$
ins(c, G_1) = \begin{cases} c \cup G_1 & \text{se } c \cup G_1 \in AFC, \\ c & \text{caso contrário.} \end{cases}
$$

Esta função é modelada pela seguinte definição:

$$
\begin{array}{|l}
ins : C \times \mathbb{P}(G) \pfun C \\
\hline
[insSimDef] \\
\forall c : C;\ G_1 : \mathbb{P}(G) \mid c \cup G_1 \in AFC \bullet ins(c, G_1) = c \cup G_1 \\
[insNaoDef] \\
\forall c : C;\ G_1 : \mathbb{P}(G) \mid c \cup G_1 \notin AFC \bullet ins(c, G_1) = c
\end{array}
$$

A operação de supressão del remove um conjunto de genes do cromossomo de origem.

Definição 3.4.15. (Supressão) - A supressão é uma função del do seguinte tipo:

$$
del : C \times \mathbb{P}(G) \to C
$$

$$
del(c, G_1) = \begin{cases} c - G_1 & \text{se } c - G_1 \in AFC, \\ c & \text{caso contrário.} \end{cases}
$$

A definição axiomática que define esta função é apresentada abaixo:

$$
\begin{array}{|l}
del : C \times \mathbb{P}(G) \pfun C \\
\hline
[delSimDef] \\
\forall c : C;\ G_1 : \mathbb{P}(G) \mid c \setminus G_1 \in AFC \bullet del(c, G_1) = c \setminus G_1 \\
[delNaoDef] \\
\forall c : C;\ G_1 : \mathbb{P}(G) \mid c \setminus G_1 \notin AFC \bullet del(c, G_1) = c
\end{array}
$$

A operação de troca $troc$ remove um conjunto de genes do cromossomo de origem e lhe adiciona outro conjunto de genes.

Definição 3.4.16. (Troca) - A troca é uma função $troc$ do seguinte tipo:

$troc : C \times \mathbb{P}(G) \times \mathbb{P}(G) \rightarrow C$

$$
troc(c, G_1, G_2) = \begin{cases}
(c \cup G_1) - G_2 & \text{se } c \cup G_1 \in AFC \wedge (c \cup G_1) - G_2 \in AFC, \\
c \cup G_1 & \text{se } c \cup G_1 \in AFC \wedge (c \cup G_1) - G_2 \notin AFC, \\
c - G_2 & \text{se } c \cup G_1 \notin AFC \wedge c - G_2 \in AFC, \\
c & \text{se } c \cup G_1 \notin AFC \wedge c - G_2 \notin AFC.
\end{cases}
$$

Assim como ins e del, $troc$ também é especificada por uma definição axiomática.

$$
\begin{array}{|l}
troc : C \times \mathbb{P}(G) \times \mathbb{P}(G) \pfun C \\
\hline
[trocSimSimDef] \\
\forall c : C;\ G_1, G_2 : \mathbb{P}(G) \mid \\
(c \cup G_1) \in AFC \wedge (c \cup G_1) \setminus G_2 \in AFC \bullet troc(c, G_1, G_2) = (c \cup G_1) \setminus G_2 \\
[trocSimNaoDef] \\
\forall c : C;\ G_1, G_2 : \mathbb{P}(G) \mid (c \cup G_1) \in AFC \wedge (c \cup G_1) \setminus G_2 \notin AFC \bullet troc(c, G_1, G_2) = c \cup G_1 \\
[trocNaoSimDef] \\
\forall c : C;\ G_1, G_2 : \mathbb{P}(G) \mid (c \cup G_1) \notin AFC \wedge c \setminus G_2 \in AFC \bullet troc(c, G_1, G_2) = c \setminus G_2 \\
[trocNaoNaoDef] \\
\forall c : C;\ G_1, G_2 : \mathbb{P}(G) \mid (c \cup G_1) \notin AFC \wedge c \setminus G_2 \notin AFC \bullet troc(c, G_1, G_2) = c
\end{array}
$$

Lema 3.4.17. $\forall c : C;\ G_1, G_2 : \mathbb{P}(G) \bullet troc(c, G_1, G_2) = del(ins(c, G_1), G_2)$

Este corolário estabelece que as ações da função de inserção e supressão podem ser vistas como casos particulares da ação da função de troca.

Definição 3.4.18. (Mutação) - A mutação é um predicado $mut \subseteq \mathbb{P}(P)$, tal que:
$mut(c_1) = \{c_2 \mid \exists G_1, G_2 : \mathbb{P}(G) ((\#G_1 \leq \#c_1 \text{ div } 2) \wedge (\#G_2 \leq \#c_1 \text{ div } 2) \wedge (troc(c_1, G_1, G_2) = c_2)$
$\wedge (adapt(c_2), adapt(c_1)) \in maior\mathbb{Q}\}$

A restrição $(adapt(c_2), adapt(c_1)) \in maior\mathbb{Q}$ garante que todo cromossomo-mutante é mais adaptado do que o cromossomo que lhe deu origem. E a limitação do número de genes que podem ser alterados do cromossomo no cromossomo-mutante em cinqüenta por cento do tamanho do cromossomo fornecido deve-se ao fato de que se as mutações ocorridas em um cromossomo de uma dada espécie forem muito grandes, então este cromossomo seria repelido pelos cromossomos da sua espécie, por não ser considerado mais um igual a estes. Em Z, a operação de mutação é especificada como:

$$mut : C \nrightarrow P$$

$[mutDef]$
$\forall c_1 : C \bullet mut(c_1) = \{c_2 : C \mid \forall G_1, G_2 : \mathbb{P}(G) \mid \#G_1 \leq \#c_1 \text{ div } 2 \wedge \#G_2 \leq \#c_1 \text{ div } 2 \bullet$
$c_2 = troc(c_1, G_1, G_2) \wedge (adapt(c_2), adapt(c_1)) \in maior\mathbb{Q}\}$

3.5 Ambiente

Um algoritmo genético opera sobre populações de cromossomos que evoluem de acordo com as características de um ambiente *A*. Um ambiente *A* é uma 8-tupla $\langle P, \mathbb{P}(P), Rq, AFG, AGC, Tx, \Sigma, P_0 \rangle$, onde:

- P é a população,
- $\mathbb{P}(P)$ é o conjunto potência de P,
- Rq é o conjunto dos requisitos (características expressas através de fórmulas numa linguagem de primeira ordem) do problema que influenciam a genealogia da população P,

- AFG é o conjunto de axiomas de formação dos genes dos cromossomos da população P,
- AFC é o conjunto de axiomas de formação dos cromossomos da população P e
- Tx é o conjunto de pares de cromossomos (x, y), onde x é um cromossomo construído a partir do cromossomo y, pela ação da operação de cruzamento ou mutação, registrando desta forma a genealogia dos cromossomos pertencentes às populações geradas pelo GAADT durante a sua execução,
- Σ é o conjunto de operadores genealógicos que atuam sobre a população P,
- P_0 é uma sub-população pertencente a $\mathbb{P}(P)$, chamada de população inicial, com no mínimo um cromossomo.

O processo de evolução darwinista, segundo o qual todas as espécies desenvolveram-se a partir de outras espécies, pela transmissão hereditária de pequenas variações, em sucessivas gerações, resultando na sobrevivência das espécies que melhor adaptaram-se ao ambiente [34], é induzido pelas alterações ambientais produzidas pela natureza. Este papel desempenhado pela natureza, na visão de evolução darwinista, aqui será representado pelo GAADT, que é quem submete os cromossomos de uma população à ação dos requisitos do problema Rq, resultando assim na geração de novos cromossomos a partir daqueles já existentes.

3.6 Algoritmo

O GAADT é uma função $GAADT$ que recebe a população P_0 e, depois de submetê-la à simulação de um processo evolutivo, devolve uma população P_t. Os cromossomos da população P_t são os cromossomos das populações P_0,P_1,...,P_{t-1} que ainda satisfazem os requisitos do problema Rq, ou então são novos cromossomos resultantes da ação genealógica das operações de cruzamento e mutação sobre os cromossomos da população P_{t-1} que apresentam adaptação maior do que a adaptação dos cromossomos que lhes deram origem. Diz-se então que a população P_t evoluiu da população P_0.

Os cromossomos das populações P_0,P_1,...,P_{t-1} que não mais satisfaçam os requisitos do problema Rq não participarão da construção da população P_t, podendo ser assim entendidos como fazendo parte da população de cromossomos "mortos", que não figurarão entre os cromossomos da população P_t e das populações seguintes manipuladas pela função $GAADT$. Não obstante, tais cromossomos serão recuperados pela análise da taxonomia Tx dos cromossomos da população atual para evitar que eles apareçam novamente nas próximas

iterações da função $GAADT$. Esta restrição atende ao entendimento do processo de evolução darwinista, que não contempla a possibilidade de uma espécie extinta voltar a aparecer num outro momento futuro.

Antes de ser apresentada uma definição para a função $GAADT$, deve-se observar a necessidade de se estabelecer um critério de preservação sobre a população atual P_t, para orientar o corte dos cromossomos que não devem figurar nas populações P_{t+1},P_{t+2},.... Na definição da função $GAADT$, este ponto de corte será representado por um predicado unário p_{corte} pertencente ao conjunto de requisitos do problema Rq, que atua sobre os cromossomos de P_t.

$$p_{corte} : P \nrightarrow P$$

$$[PontoCorteDef]$$
$$\forall P_1 : P \bullet p_{corte}(P_1) \subseteq P_1$$

Por exemplo, para o problema do caixeiro viajante, o ponto de corte poderia selecionar somente os cromossomos com grau de adaptação maior ou igual ao valor da adaptação média da população. A adaptação média da população é obtida dividindo-se a soma da adaptação de todos os cromossomos da população ($adaptTotal$) pelo número de cromossomos desta população.

$$adaptTotal : P \nrightarrow \mathbb{Q}^+$$

$$[adaptTotalDef]$$
$$adaptTotal(\emptyset) = (0,1) \land$$
$$(\forall P_1 : \mathbb{P}(P) \bullet \forall c : P_1 \bullet adaptTotal(P_1) = adicao\mathbb{Q}(adapt(c), adaptTotal(P_1 \setminus \{c\})))$$

$$adaptMedia : P \nrightarrow \mathbb{Q}^+$$

$$[adaptMediaDef]$$
$$adaptMedia(\emptyset) = (0,1) \land (\forall P_1 : \mathbb{P}(P) \bullet adaptMedia(P_1) = divisao\mathbb{Q}(adaptTotal(P_1), (\#P_1, 1)))$$

$$p_{corte} : P \nrightarrow P$$

$$[PontoCorteDef]$$
$$\forall P_1 : P \bullet p_{corte}(P_1) = \{c : P_1 \mid (adapt(c), adaptMedia(p)) \in maior\mathbb{Q} \vee (adapt(c), adaptMedia(P_1)) \in mesmo\mathbb{Q}\}$$

Os critérios de parada adotados pela função $GAADT$ são: a) o número máximo de iterações desejadas, b) valor da adaptação dos cromossomos considerado satisfatório para o resultado do problema em análise. Estes critérios também fazem parte do conjunto de requisitos do problema Rq.

Definição 3.6.1. (GAADT) - O GAADT é uma função $GAADT$ do seguinte tipo:

$$GAADT : A \rightarrow A$$
$$GAADT(P_t) = \begin{cases} P_{otm} & \text{se } P_{otm} = \{c \mid \forall c : P_t \, (adapt(c) \geq k)\} \neq \emptyset, \\ P_{t+1} & \text{se } t + 2 = T, \\ GAADT(P_{t+1}) & \text{caso contrário.} \end{cases}$$

onde $P_{t+1} = cruz(a, b) \cup mut(c) \cup p_{corte}(P_t)$ com e $a, b, c \in P_t$, P_0 é a população inicial considerada, $k \in K$ é um valor imposto pelo ambiente A, como critério de aceitação de cromossomos em P_t que satisfazem o problema e $T \in \mathbb{N}$ é um número dado como critério de satisfação do número de iterações.

No processo acima, a primeira e a segunda opções de saída são condições de parada. A segunda ocorrerá garantidamente, se eventualmente a primeira não ocorrer. Também deve ser observado que para qualquer entrada P_t, o processo $GAADT$ dá uma saída bem determinada. Isto significa que o $GAADT$ é um algoritmo e desta forma um procedimento correto.

A especificação desta função requer a definição dos elementos população inicial, número máximo de iterações e o valor da adaptação do cromossomo considerada satisfatória para a busca atual.

$$P_0 : \mathbb{P}(P)$$
$$T : \mathbb{Z}$$
$$k : \mathbb{Q}$$

Por exemplo, no caso do problema do caixeiro viajante a população inicial pode ser composta somente com a rota que começa na cidade $CidadeA$, passa pelas cidades $CidadeB$, $CidadeC$ e $CidadeD$, nesta ordem, e termina na cidade $CidadeA$. Como não se sabe, a *priori*, qual é o valor da adaptação da menor rota, então estabelece-se que este valor deve ser zero, o que é inatingível para este problema considerando o valor da distância entre as cidades (ver Tabela 3.1). Assim, o algoritmo só poderá parar pela condição de número máximo de iterações, que neste caso é 100.

$$
\begin{array}{|l}
P_0 : \mathbb{P}(P) \\
T : \mathbb{Z} \\
k : \mathbb{Q} \\
\hline
P_0 = \{\{\langle CidadeA, CidadeB\rangle, \langle CidadeB, CidadeC\rangle, \langle CidadeC, CidadeD\rangle, \langle CidadeD, CidadeA\rangle\}\} \land \\
T = 100 \land \\
k = (0, 1)
\end{array}
$$

O estado do sistema é composto pela população de cromossomos vivos P_{atual}, pela população de cromossomos gerados por cruzamento P_{cruz}, pela população de cromossomos gerados por mutação P_{mut}, pela relação taxonômica dos cromossomos da população atual com seus ancestrais Tx e pelo contador do número de iterações t. As propriedades invariantes deste estado é que os conjuntos $MACHO$ e $FEMEA$ são subconjuntos

da população atual, e que não existe dois cromossomos equivalentes na população atual.

$$
\begin{array}{|l}
\hline
StateGAADT \\
\hline
Tx : C \leftrightarrow C \\
P_{atual} : P \\
P_{cruz} : P \\
p_{mut} : P \\
t : \mathbb{Z} \\
MACHO : P \\
FEMEA : P \\
\hline
MACHO \subseteq P_{atual} \land \\
FEMEA \subseteq P_{atual} \land \\
(\forall c_1, c_2 : P_{atual} \bullet (c_1, c_2) \notin \equiv_C)) \\
\hline
\end{array}
$$

Quando o sistema é inicializado, a população de cromossomos vivos possui o valor P_0, a relação taxonômica dos cromossomos desta população P_0 vazia e o contador do número de iterações tem o valor zero para indicar que nenhuma iteração da função $GAADT$ foi realizada. A população de cromossomos gerados, por cruzamento e mutação, está vazia esperando que uma chamada das funções $cruz$ e mut seja executada.

$$
\begin{array}{|l}
\hline
InitGAADT \\
\hline
StateGAADT' \\
\hline
Tx' = \emptyset \land \\
p'_{atual} = P_0 \land \\
P'_{cruz} = \emptyset \land \\
P'_{mut} = \emptyset \land \\
t' = 0 \land \\
MACHO' = sel(p_{corte}(P_0), M) \land \\
FEMEA' = sel(p_{corte}(P_0), F) \\
\hline
\end{array}
$$

O esquema $Cruzamento$ especifica a construção de novos cromossomos pela ação da função $cruz$ sobre os cromossomos pertencentes ao conjunto de cromossomos da população atual que satisfazem o critério p_{corte},

que também pertencem ao conjunto $MACHO$ ou $FEMEA$.

$$
\begin{array}{|l}
\hline
Cruzamento \\
\hline
\Delta StateGAADT \\
\hline
Tx' = Tx \land \\
P'_{atual} = P_{atual} \land \\
P'_{cruz} = \{c_1 : C \mid (\exists c_2 : MACHO;\ c_3 : FEMEA \bullet c_1 \in cruz(c_2, c_3)) \land \\
(\forall c_4, c_5 : C \mid (c_4, c_5) \in Tx \bullet (c_4, c_1) \notin \equiv_C \land (c_5, c_1) \notin \equiv_C)\} \land \\
P'_{mut} = P_{mut} \land \\
t' = t \land \\
MACHO' = MACHO \land \\
FEMEA' = FEMEA \\
\hline
\end{array}
$$

O esquema $Mutacao$ descreve o mecanismo de formação de novos cromossomos pela ação do predicado mut sobre os cromossomos da população atual que não satisfazem ao critério p_{corte} especificado.

$$
\begin{array}{|l}
\hline
Mutacao \\
\hline
\Delta StateGAADT \\
\hline
Tx' = Tx \land \\
P'_{atual} = P_{atual} \land \\
P'_{cruz} = P_{cruz} \land \\
P'_{mut} = \{c_1 : C \mid (\forall c_2 : P_{atual} \setminus p_{corte}(P_{atual}) \bullet c_1 \in mut(c_2)) \land \\
(\exists c_3, c_4 : C \mid (c_3, c_4) \in Tx \bullet (c_1, c_3) \notin \equiv_C \land (c_1, c_4) \notin \equiv_C)\} \land \\
t' = t \land \\
MACHO' = MACHO \land \\
FEMEA' = FEMEA \\
\hline
\end{array}
$$

O esquema $Iteracao$ retrata a construção da população P_{t+1}, a inclusão da relação taxonômica entre os cromossomos população P_{t+1} e os cromossomos da população P_t, o incremento do contador de iteração e a

preparação das populações P_{cruz} e P_{mut} para a próxima iteração do $GAADT$.

$$
\begin{array}{|l}
\hline
Iteracao \\
\Delta StateGAADT \\
\hline
Tx' = Tx \cup \{c_1 : P_{atual};\ c_2 : P_{cruz} \mid \exists c_3 : P_{atual} \bullet c_2 \subseteq cruz(c_1, c_2)\} \\
\cup \{c_1 : P_{atual};\ c_2 : P_{mut} \mid c_2 \subseteq mut(c_1)\} \wedge \\
P'_{atual} = p_{corte}(P_{atual}) \cup P_{cruz} \cup P_{mut} \wedge \\
P'_{cruz} = \emptyset \wedge \\
P'_{mut} = \emptyset \wedge \\
t' = t + 1 \wedge \\
MACHO' = MACHO \wedge \\
FEMEA' = FEMEA \\
\hline
\end{array}
$$

O esquema $TerminaPorAdaptacao$ especifica quando a função $GAADT$ termina sua execução pelo critério de descrição das características de uma população ótima para o problema.

$$
\begin{array}{|l}
\hline
TerminaPorAdaptacao \\
\Xi StateGAADT \\
\hline
\exists c : P_{atual} \bullet (adapt(c), k) \in mesmo\mathbb{Q} \vee (adapt(c), k) \in maior\mathbb{Q} \\
\hline
\end{array}
$$

Já o esquema $TerminaPorTempo$ descreve a condição de parada da função $GAADT$ pelo critério de número máximo de iterações.

$$
\begin{array}{|l}
\hline
TerminaPorTempo \\
\Xi StateGAADT \\
\hline
t' = T \\
\hline
\end{array}
$$

O esquema $Termina$ retrata as condições de parada da função $GAADT$ através da disjunção dos esquemas $TerminaPorAptacao$ e $TerminaPorTempo$.

$$Termina \widehat{=} \quad TerminaPorAdaptacao \vee TerminaPorTempo$$

Finalmente a função $GAADT$ é modelada como a disjunção das condições de parada com a composição entre a conjunção das operações genealógicas com a construção da próxima população.

$$GAADT \widehat{=} \quad Termina \vee ((Cruzamento \wedge Mutacao) \; {}^{0}_{9} \; Iteracao)$$

Capítulo 4

Convergência do Algoritmo Genético Baseado em Tipos Abstratos de Dados

4.1 Introdução

Holland via o algoritmo genético como um algoritmo de busca, onde os cromossomos são amostras do espaço de busca e os operadores são mecanismos capazes de direcionar o algoritmo para melhores amostras, através da preservação das informações codificadas nos cromossomos mais adaptados realizada pela operação de cruzamento, e da investigação de novas amostras geradas pela operação de mutação. Para descrever formalmente o comportamento do seu algoritmo, Holland desenvolveu a teoria do *schema* que adota o *schema* como uma forma para representar os subespaços do espaço de busca que contém os cromossomos da população p.

Definição 4.1.1. (*Schema*) - Um $schema$ é uma m-upla $s = (x_1, ..., x_m)$, tal que todo elemento x_j de s ou é uma variável $v_1, v_2, ..., v_w$, com $w \leq m$, ou é uma constante "0" ou "1".

Dado um $schema$ $s = (x_1, ..., x_m)$ o subespaço do espaço de busca representado por s, denotado por $\mathbb{S}_s$, é formado por todos os vetores obtidos pela substituição das variáveis existentes em s pelas constantes "0" e "1". De onde se conclui que o número de elementos pertencentes ao subespaço $\mathbb{S}_s$ é 2^w, onde w é o número de variáveis de s.

Definição 4.1.2. (Subespaço representado por um *schema*) - Seja $s = (x_1, ..., x_m)$ um $schema$, o subespaço representado por s é o conjunto $\mathbb{S}_s = \{\langle y_1, y_2, ..., y_m\rangle \mid \forall j : \{1, ..., m\}\, (j \in const(s) \rightarrow y_j = x_j) \wedge (j \in var(s) \rightarrow y_j = 0 \vee y_j = 1)\}$, onde $const$ é uma função que recebe um $schema$ e retorna o conjunto das

posições onde uma constante ocorre no $schema$ fornecido, e var é uma função que recebe um *schema* e retorna o conjunto das posições onde uma variável ocorre no $schema$ fornecido.

A cada $schema\ s$ de uma população p um valor de adaptação é associado para informar o quanto o subespaço representado pelo $schema\ s$ é um subespaço interessante para o problema a ser resolvido. Por convenção o conjunto dos $schemata$ de uma população p é denotado de $\mathbb{S}_p$.

Definição 4.1.3. (Adaptação do *schema*) - A adaptação de um $schema$ é uma função ϕ, do seguinte tipo:

$\phi : \mathbb{S}_p \times \mathcal{P} \rightarrow \mathbb{R}$

$\phi(s,p) = \frac{\sum_{j=1}^{n} f(p_j)}{n}$

De posse da representação dos subespaços manipulados pelo algoritmo genético e do grau de adaptação destes subespaços ao problema em foco, Holland mostrou como os subespaços de uma população na iteração t são menos adaptados do que os subespaços da população na iteração $t+1$, através da análise do comportamento probabilístico deste algoritmo.

No algoritmo de Holland a primeira operação aplicada à população em qualquer iteração t é a seleção elitista, que determina os *schemata* a serem investigados, através do armazenamento dos cromossomos da população p que pertencem a estes $schemata$ na população intermediária $\check{p}$. O número de cromossomos na população $\check{p}$ que pertencem ao *schemta* s da população p é $\xi(s,\check{p}) = \xi(s,p) \times h \times \phi(s,p)$, onde $\xi(s,p)$ é o número de cromossomos que pertencem ao $schema\ s$ na população p e $\phi(s,p)$ é a adaptação do $schema\ s$ na população p (Definição 4.1.3).

Após a escolha dos cromossomos da população habilitados para gerar descendentes na próxima iteração do algoritmo, as operações de cruzamento de um ponto de corte, mutação por complemento e mutação por inversão são aplicadas sobre os cromossomos da população $\check{p}$. A probabilidade do $schema\ s$ ser preservado pela ação da operação de cruzamento é $p_{\triangle}(s) \geq 1 - prob_c \times \frac{\triangle(s)}{m-1}$, onde $\triangle\ (s)$ é a distância entre a primeira e a última ocorrência de uma constante no *schema* s. O termo $\frac{\triangle(s)}{m-1}$ desta inequação lembra que a única possibilidade do cruzamento entre os cromossomos $\langle a_1, a_2, ..., a_m\rangle$ e $\langle b_1, b_2, ..., b_m\rangle$, pertencentes ao conjunto de *schema* $\mathbb{S}_a$ e $\mathbb{S}_b$ distintos, tal que $\mathbb{S}_a \cap \mathbb{S}_b = \emptyset$ e $s \in \mathbb{S}_a \cup \mathbb{S}_b$, preservar um $schema\ s$ é fazer com que o ponto de corte j do cruzamento não coincida com uma das posições entre a primeira e a última ocorrência de uma constante neste $schema$. O sinal "$\geq$" na formula é usado para lembrar que se $s \in \mathbb{S}_a \cap \mathbb{S}_b$, então todos os cromossomos gerados devem pertencer a s.

A probabilidade do $schema\ s$ ser preservado pela ação da operação de mutação por complemento é $p_{\mho}(s) = (1 - prob_m)^{m-w}$, onde w é o número de variáveis no $schema$. Esta equação diz que a única possibilidade da mutação por complemento sobre um cromossomo $\langle a_1, a_2, ..., a_m \rangle$, pertencente ao conjunto de $schema\ \mathbb{S}_a$, tal que $s \in \mathbb{S}_a$, preservar o $schema\ s$ é fazer com que os pontos de mutação não coincidam com as posições das constantes neste $schema$.

A probabilidade do $schema\ s$ ser preservado pela ação da operação de mutação por inversão é $p_{\triangledown}(s) \geq 1 - prob_i \times \frac{m-w}{w}$, onde w é o número de variáveis no $schema\ s$. Esta inequação estabelece que a única possibilidade da mutação por inversão sobre um cromossomo $\langle a_1, a_2, ..., a_m \rangle$, pertencente ao conjunto de $schema\ \mathbb{S}_a$, tal que $s \in \mathbb{S}_a$, preservar o $schema\ s$ é fazer com que as posições a terem seu conteúdo alterado não coincidam com as posições das constantes neste $schema$. O sinal "$\geq$" na inequação é usado para estabelecer que, se uma constante for trocada por ela mesma, então o $schema\ s$ é preservado.

Por fim, constrói-se a nova população pela substituição da população atual por uma nova população p formada pelos $n-k$ cromossomos mais adaptados da população p atual e pelos k cromossomos mais adaptados da população $\dot{p}$. O número de cromossomos pertencentes ao *schema* s na nova população p é $\xi(s,p) \geq \xi(s,p) \times (n-k) \times \frac{\phi(s,p)}{\sum_{x \in p} f(x)} + \xi(s,\check{p}) \times k \times \frac{\phi(s,\dot{p})}{\sum_{x \in \dot{p}} f(x)} \times p_{\triangle}(s) \times p_{\mho}(s) \times p_{\triangledown}(s)$, onde $\xi(s,p) \times (n-k) \times \frac{\phi(s,p)}{\sum_{x \in p} f(x)}$ é o número de cromossomos pertencentes ao $schema\ s$ entre os $n-k$ cromossomos mais adaptados da população p atual e $\xi(s,\check{p}) \times k \times \frac{\phi(s,\dot{p})}{\sum_{x \in \dot{p}} f(x)} \times p_{\triangle}(s) \times p_{\mho}(s) \times p_{\triangledown}(s)$ é número de cromossomos pertencentes ao $schema\ s$ entre os k cromossomos mais adaptados da população $\dot{p}$.

A teoria do *schema* é aplicável somente a uma simples iteração, servindo assim somente para dar uma pequena intuição da convergência do algoritmo genético por considerar a sobreposição dos subespaços investigados por outros subespaços mais adaptados do que os seus antecessores. Uma forma de tratar a convergência do algoritmo genético foi proposta por Goldberg [58] sob o nome de hipótese do bloco de construção, segundo a qual um algoritmo genético busca cromossomos com um desempenho ótimo pela justaposição de *schemata* pequenos, de baixa ordem e com alta adaptação denominados de blocos de construção.

Definição 4.1.4. (Hipótese do bloco de construção) - Um algoritmo genético converge para uma população formada por cromossomos pertencentes a *schemata* pequenos, de baixa ordem e com adaptação acima da média dos *schemata* existentes na população atual.

Como o próprio Goldberg [58, 82] observa, para que a hipótese do bloco de construção seja bem sucedida

é necessário que as informações codificadas no cromossomo não sejam dependentes entre si, já que este tipo de relacionamento não pode ser capturado pelo $schema$. A presença de dependência entre as informações codificadas no cromossomo (epistasis) engana o algoritmo genético, fazendo o algoritmo convergir para um bloco de construção que contém parte destas informações [69].

Szalas e Michalewics [157] apresentam uma outra abordagem para explicar a convergência de um algoritmo genético baseada no teorema do ponto fixo de Banach [9], que trabalha com mapeamentos contrativos em espaços métricos e afirma que qualquer mapeamento deste tipo tem um único ponto fixo, isto é, o algoritmo genético converge sempre para uma mesma população independente da população inicial considerada.

Um espaço métrico completo é um par ordenado (E, d), onde E é o conjunto de todas as populações possíveis de serem manipuladas pelo algoritmo e $d : E \times E \rightarrow \mathbb{R}$ é uma função que permite medir a distância entre quaisquer duas populações, tal que as seguintes propriedades são satisfeitas:

- $\forall e_1$,$e_2 \in E\ (e_1 = e_2 \Leftrightarrow d(e_1, e_2) = 0)$,
- $\forall e_1$,$e_2 \in E\ (e_1 \neq e_2 \Leftrightarrow d(e_1, e_2) > 0)$,
- $\forall e_1$,$e_2 \in E\ (d(e_1, e_2) = d(e_2, e_1))$,
- $\forall e_1$,$e_2, e_3 \in E\ (d(e_1, e_2) + d(e_2, e_3) \geq d(e_1, e_3))$ e
- $\forall e_1$, e_2,...,e_w seqüência de Cauchy $\exists r_1 \in (0, 1]\,(\forall j_1, j_2 \in \{1, 2, ..., w\}\,(d(e_{j_1}, e_{j_2}) < r_1))$, tem um limite quando w tende para o infinito.

Na visão de Szalas e Michalewics, um algoritmo genético m é visto como uma transformação contrativa entre populações, para as quais existem espaços métricos completos, ou seja, $\exists r_1 \in [0, 1]\,(\forall e_1, e_2 \in E\,(d(m(e_1), m(e_2)) \leq r_1 \times d(e_1, e_2)))$.

Teorema 4.1.5. *(Teorema de Banach) - Seja (E, d) um espaço métrico completo e $m : E \leftrightarrow E$ um mapeamento contrativo. Então este mapeamento tem um ponto fixo $\overline{e} \in E$, tal que $lim_{j\rightarrow\infty} m(e_j) = \overline{e}$, isto é $m(\overline{e}) = \overline{e}$.*

Os problemas para os quais existem mais de um resultado possível não podem ser avaliados segundo esta abordagem, porque para tais problemas existem mais de um ponto fixo, o que é contrário ao pressuposto pelo teorema de Banach de que o ponto fixo é único localmente.

As dificuldades de propor uma teoria para explicar o comportamento de um algoritmo genético e conseqüentemente de sua convergência esbarrar nas condições topológicas especiais contempladas por estes mecanismos computacionais. O fato do ambiente A ser um ambiente evolutivo significa que, com exceção da população inicial, todas as outras populações P_{t+1}, com $t \geq 0$, são formadas a partir da população atual P_t pelo nascimento, preservação e morte de cromossomos. Por isto, a taxonomia populacional de um algoritmo genético pode ser vista como um encaixamento parcial da população P_t na população P_{t+1} (Figura 4.1).

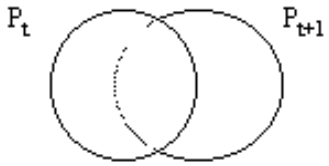


Figura 4.1: Encaixe das populações

Esta percepção do encaixe taxonômico das populações manipuladas por um algoritmo genético conduz à seguinte intuição sobre a sua convergência: seja P_{res} uma subpopulação de P_t, com $t \leq T$ para algum T suficientemente grande, formada por todos cromossomos que atendem a uma condição de adaptação k desejada, tal que a população P_t é quase a população P_{res}, no sentido de que a presença dos cromossomos que diferenciariam P_t de P_{res} é desprezível. Diz-se então que a população P_t converge para P_{res} [43, 46].

A dinâmica dos encaixes taxônomicos realizada durante a convergência de um algoritmo genético permite a observação das seguintes situações:

1. a população desejada P_{res} está contida quase completamente em P_0 - assim a seqüência P_0,P_1,...,P_t, terá uma convergência por redução (diminuição) da população P_0 até uma população P_t que é quase a população P_{res};

2. a população desejada P_{res} está contida parcialmente em P_0 - desse modo, a seqüência P_0, P_1,...,P_t evoluirá (crescendo) para uma população P_t que contém a população P_{res} e depois será reduzida como no primeiro caso;

3. a interseção entre as populações P_0 e P_t é vazia, então a seqüência P_0,P_1,...,P_t, evoluiria até produzir um P_t que contenha P_{res} e depois se comportaria como no primeiro caso.

O uso do pronome relativo “quase” nestas situações se justifica porque o encaixe taxonômico das populações não se dá completamente, muito embora no limite, quando $t = T$, ele tenda a ser praticamente completo (Figura 4.2). Este comportamento se deve à constante alteração causada pelo surgimento (nascimento) de novos cromossomos gerados pela ação dos operadores genéticos de cruzamento e mutação, ou ao contrário, pela supressão (morte) dos cromossomos considerados não adaptados ao ambiente durante o processo de evolução da população.

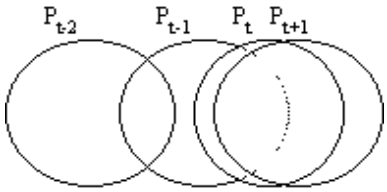


Figura 4.2: Encaixe quase completo das populações

A teoria de processos evolutivos, apresentada no restante deste capítulo, foi desenvolvida com o intuito de descrever a convergência dos algoritmos genéticos dentro de ambientes evolutivos independente da existência ou não de epistasis e do número de resultados de qualidade satisfatória possíveis para o problema em análise. Para tanto esta teoria baseia-se simplesmente no conceito de adaptação média da população trabalhada pelo algoritmo genético e no teorema do ponto fixo de Tarski [155].

4.2 Esboço de uma Teoria de Processos Evolutivos

Dado um ambiente evolutivo A e duas populações P_1 e P_2 pertencentes a $\mathbb{P}(P)$, diz-se que estas populações são equivalentes evolutivamente se elas apresentam o mesmo valor de adaptação média. Isto porque do ponto de vista da metáfora de Darwin para resolver problemas, as populações com esta propriedade têm a mesma relevância já que a adaptação média é o ponto de referência natural (p_{corte}) para a seleção dos “melhores” cromossomos segundo o perfil dos resultados desejados.

Definição 4.2.1. (Equivalência Evolutiva) - A equivalência evolutiva é uma relação $\equiv_E$, do seguinte tipo:
$\equiv_E$: $\mathbb{P}(P) \times \mathbb{P}(P)$, tal que
$P_1 \equiv_E P_2$, se e somente se, $adapt_m(P_1) = adapt_m(P_2)$

O uso da adaptação média como ponto de referência também pode ser utilizado para definir uma ordem sobre as populações construídas durante a execução de um algoritmo genético. Assim a simples constatação de que a adaptação média de uma população P_1 é inferior a adaptação média de uma população P_2, indica que a população P_2 possui provavelmente cromossomos mais próximos dos resultados desejados para o problema em questão do que a população P_1. Isto independentemente das populações P_1 e P_2 possuírem ou não cromossomos comuns, ou seja, do ponto de vista da solução do problema só a adaptação média interessa como critério para direcionamento a busca de um algoritmo genético para uma população que contém os cromossomos cuja adaptação atende as condições desejadas.

Definição 4.2.2. (Ordem Evolutiva) - A ordem evolutiva é uma relação $\subseteq_E$ do seguinte tipo:
$\subseteq_E$: $\mathbb{P}(P) \times \mathbb{P}(P)$, tal que
$P_1 \subseteq_E P_2$, se e somente se, $adapt_m(P_1) \leq adapt_m(P_2)$

Deve-se observar que "$\leq$" é uma relação de ordem sobre o corpo K referido na definição do grau de adaptação de um gene (Definição.3.4.1), condição esta que será herdada pela relação de equivalência evolutiva, como será mostrado.

Proposição 4.2.3. *$S(P) = \langle \mathbb{P}(P), \subseteq_E \rangle$ é um sistema ordenado, isto é, a relação $\subseteq_E$ é uma relação de ordem parcial sobre $\mathbb{P}(P)$.*

Prova: Com efeito são verificadas as seguintes propriedades:

1. $\forall P_1 \in \mathbb{P}(P)(P_1 \subseteq_E P_1)$ - A prova é direta da definição.

2. $\forall P_1, P_2 \in \mathbb{P}(P)((P_1 \subseteq_E P_2 \wedge P_2 \subseteq_E P_1) \implies P_1 \equiv_E P_2)$ - Se $P_1 \subseteq_E P_2$, então pela definição de ordem evolutiva tem-se que $adapt_m(P_1) \leq adapt_m(P_2)$. Por outro lado, de forma similar tem-se que $adapt_m(P_2) \leq adapt_m(P_1)$. Como "$\leq$" é uma relação de ordem sobre o corpo K, então se obtém que $adapt_m(P_1) = adapt_m(P_2)$. Logo pela definição de equivalência evolutiva (Definição 4.2.1), tem-se que $P_1 \equiv_E P_2$.

3. $\forall P_1, P_2, P_3 \in \mathbb{P}(P)((P_1 \subseteq_E P_2 \wedge P_2 \subseteq_E P_3) \implies P_1 \subseteq_E P_3)$ - Esta propriedade é imediata da definição de ordem evolutiva (Definição 4.2.2) e das hipóteses que $adapt_m(P_1) \leq adapt_m(P2) \leq adapt_m(P_3)$. De onde se conclui por transitividade que $P_1 \subseteq_E P_3$.□

Um algoritmo genético assim como qualquer outro algoritmo da computação evolutiva atua sobre populações de cromossomos que vão se entrelaçando através das relações de parentesco, as quais são geradas por procedimentos computacionais que simulam convenientemente a Teoria de Darwin [46, 60, 79]. A conveniência referida é no sentido de que os algoritmos genéticos têm como objetivo a solução de problema [111] e não necessariamente reproduzir a evolução das espécies como ela ocorre na natureza [40]. A idéia é que os algoritmos genéticos façam parte do que será chamado, de agora em diante, de família de processos evolutivos.

Definição 4.2.4. (Processo Evolutivo) - Um processo evolutivo é uma função PE do seguinte tipo:
$PE : \mathbb{P}(P) \rightarrow \mathbb{P}(P)$, tal que
$\forall P_j \in \mathbb{P}(P)(P_j \subseteq_E PE(P_j))$.

Esta definição sugere que um processo evolutivo é fundamentalmente uma estratégia de melhoramento da adaptação média da população. Isto, dentro da concepção de que os cromossomos de uma população são representações computacionais dos possíveis resultados do problema e a adaptação é uma medida usada para identificar quão próximo um dado cromossomo está dos resultados desejados para um problema.

Os processos evolutivos que interessam para a descrição da convergência de um algoritmo genético são aqueles para os quais existe uma população $P_e \in \mathbb{P}(P)$, denominada de população estagnada (ponto fixo), tal que a variação da adaptação média da população P_e, obtida em decorrência de aplicações sucessivas de um processo evolutivo PE sobre a população inicial P_0 é tão pequena, que o valor da adaptação média desta população praticamente não se altera.

Definição 4.2.5. (População Estagnada) - Uma população $P_e \in \mathbb{P}(P)$ é dita estagnada com respeito a um processo evolutivo PE, se e somente se,
$\forall P_e \in \mathbb{P}(P)(P_e \subseteq_E PE(P_e) \implies adapt_m(P_e) = adapt_m(PE(P_e)))$.

Proposição 4.2.6. *Se P_e é uma população estagnada e PE é um processo evolutivo sobre $\mathbb{P}(P)$, então $PE(P_e) \equiv_E P_e$.*

Prova: Imediata das Definições 4.2.1, 4.2.4 e 4.2.5. □

A proposição acima expressa que a população estagnada é invariante com respeito a um processo evolutivo, isto é, a população P_e é um ponto de estagnação da dinâmica do processo evolutivo PE. Esta invariância

significa que os cromossomos da população estagnada alcançaram a melhor adaptação possível, sendo os cromossomos mais adaptados da população estagnada (critério k da Definição 3.6.1) os resultados desejados do problema para o qual o processo evolutivo PE é adotada como solução.

Um processo evolutivo é monotônico se para quaisquer populações P_1 e P_2 pertencentes a $\mathbb{P}(P)$, com a população P_1 precedendo a população P_2 de acordo com a relação ordem evolutiva, tem-se que a população $PE(P_1)$ precede a população $PE(P_2)$ segundo a mesma relação.

Definição 4.2.7. (Monotonicidade) - Um processo evolutivo PE é dito monotônico (Definição 4.2.7, se e somente se, $\forall P_1, P_2 \in \mathbb{P}(P)(P_1 \subseteq_E P_2 \implies PE(P_1) \subseteq_E PE(P_2))$.

As populações geradas por processos evolutivos mantêm relações de interseção evolutiva, união evolutiva e diferença evolutiva entre si em função da adaptação de seus cromossomos e da sua adaptação média, como definido a seguir.

Definição 4.2.8. (Relações Evolutivas) - As relações interseção evolutiva ($\cap_E$), união evolutiva ($\cup_E$) e diferença evolutiva ($-_E$) sobre $\mathbb{P}(P)$, são definidas do seguinte modo:

$P_1 \cap_E P_2 \equiv \{x \mid x \in P_1 \cap P_2 \wedge adapt(x) \leq min[adapt_m(P_1), adapt_m(P_2)]\}$

$P_1 \cup_E P_2 \equiv \{x \mid x \in P_1 \cup P_2 \wedge adapt(x) \geq max[adapt_m(P_1), adapt_m(P_2)]\}$

$P_1 -_E P_2 \equiv P_1 - \{x \mid x \in P_2 \wedge adapt(x) \leq adapt_m(P_1)\}$

Proposição 4.2.9. *A interseção evolutiva de duas populações P_1 e P_2 precede cada uma destas populações segundo a relação de ordem evolutiva, ou seja, $P_1 \cap_E P_2 \subseteq_E P_1$ e $P_1 \cap_E P_2 \subseteq_E P_2$.*

Prova: $adapt_m(P_1 \cap_E P_2) \leq adapt_m(P_1)$ e $adapt_m(P_1 \cap_E P_2) \leq adapt_m(P_2)$. □

Proposição 4.2.10. *A união evolutiva de duas populações P_1 e P_2 é precedida por cada uma destas populações segundo a relação de ordem evolutiva, ou seja, $P_1 \subseteq_E P_1 \cup_E P_2$ e $P_2 \subseteq_E P_1 \cup_E P_2$.*

Prova: $adapt_m(P_1) \leq adapt_m(P_1 \cup_E P_2)$ e $adapt_m(P_2) \leq adapt_m(P_1 \cup_E P_2)$. □

Proposição 4.2.11. *A diferença evolutiva de duas populações P_1 e P_2 é precedida por uma destas populações segundo a relação de ordem evolutiva, ou seja, $P_1 \subseteq_E P_1 -_E P_2$.*

Prova: $adapt_m(P_1) \leq adapt_m(P_1 -_E P_2)$. □

Lema 4.2.12. $P_1 \subseteq_E P_2 \implies P_1 \cap_E P_2 \equiv_E P_1$

Prova: Deve-se observar que, se $P_1 \subseteq_E P_2$, então $min[adapt_m(P_1), adapt_m(P_2)] = adapt_m(P_1)$, logo $P_1 \subseteq_E P_1 \cap_E P_2$. Pela Proposição 4.2.9 da relação de interseção evolutiva tem-se que $P_1 \cap_E P_2 \equiv_E P_1$. □

Lema 4.2.13. $P_1 \subseteq_E P_2 \implies P_1 \cup_E P_2 \equiv_E P_2$

Prova: Dual da prova do Lema 4.2.12.□

Proposição 4.2.14. *Um processo evolutivo PE é dito "monotônico", se e somente se,* $\forall P_1, P_2 \in \mathbb{P}(P)$ $(PE(P_1 \cap_E P_2) \subseteq_E PE(P_1) \cap_E PE(P_2)$.

Prova: Considere que $\forall P_1, P_2 \in \mathbb{P}(P)((P_1 \cap_E P_2 \subseteq_E P_1) \wedge (P_1 \cap_E P_2 \subseteq_E P_2))$, pela Proprosição 4.2.9 da relação de interseção evolutiva. Como por hipótese, PE é um processo evolutivo monotônico, então $PE(P_1 \cap_E P_2) \subseteq_E PE(P_1)$ e $PE(P_1 \cap_E P_2) \subseteq_E PE(P_2)$. Assim, tem-se que $PE(P_1 \cap_E P_2) \subseteq_E PE(P_1) \cap_E PE(P_2)$.

Inversamente, assuma $\forall P_1, P_2 \in \mathbb{P}(P)(PE(P_1 \cap_E P_2) \subseteq_E PE(P_1) \cap_E PE(P_2))$ por hipótese. Tem-se pela Proposição 4.2.9 que $(P_1 \cap_E P_2) \subseteq_E PE(P_1)$. Como $P_1 \subseteq_E P_2$ tem-se que $PE(P_1 \cap_E P_2) \subseteq_E PE(P_1) \cap_E PE(P_2) \subseteq_E PE(P_2)$, pela Proposição 4.2.3 de sistemas ordenados, tem-se que $P_1 \cap_E P_2 \equiv_E P_1$ e em decorrência $PE(P_1 \cap_E P_2) \equiv_E PE(P_1) \subseteq_E PE(P_2)$. □

Proposição 4.2.15. *Se PE é um processo evolutivo, então $\forall j \in \mathbb{N}$, vale que:*

$$PE(\bigcap_{Ej \in \mathbb{N}} P_j) \subseteq_E \bigcap_{Ej \in \mathbb{N}} PE(P_j).$$

Prova: Por indução finita usando a idéia da prova da Proposição 4.2.14. □

Teorema 4.2.16. *(Ponto Fixo) - Se um processo evolutivo PE é monotônico, então ele evolui para uma população estagnada, isto é, todo processo evolutivo monotônico tem ponto fixo.*

Prova: Seja $S = \bigcap_{Ej \in \mathbb{N}} \{P_j \mid PE(P_j) \subseteq_E P_j\}$. Como é evolutivo, tem-se que, $\forall X \in \mathbb{P}(P)(X \subseteq_E PE(X))$, em particular se $X = S$, então $S \subseteq_E PE(S)$. Por outro lado, tem-se que pelo fato de PE ser monotônico então $PE(\bigcap_{Ej \in \mathbb{N}} P_j) \subseteq_E \bigcap_{Ej \in \mathbb{N}} PE(P_j)$. Assim a propriedade acima também é satisfeita para S, logo $PE(S) \subseteq_E S$ e deste modo $PE(S) \equiv_E S$. □

Com este teorema será possível predicar sobre a convergência de um processo evolutivo. Tal predicação será usada para garantir que a evolução de uma população conduzida por um processo evolutivo monotônico

pára quando uma população estagnada é atingida, isto é, interpretados os elementos do domínio de um problema segundo a metáfora de Darwin, a população estagnada é constituída dos possíveis resultados do problema, fazendo assim com que a condição de solução do problema desempenhada pelo algoritmo genético seja realmente exercida.

4.3 A convergência do $GAADT$

Nesta seção será exibido o estudo da convergência do algoritmo genético $GAADT$ como está definido no Capítulo 3. Basicamente o que se pretende mostrar aqui é que o $GAADT$ é um processo evolutivo monotônico, isto é, possui uma população estagnada. Em outras palavras o $GAADT$ satisfaz as seguintes propriedades:

1. $\forall P_j \in \mathbb{P}(P)(P_j \subseteq_E GAADT(P_j)$ processo evolutivo e
2. $\forall P_1, P_2 \in \mathbb{P}(P)(P_1 \subseteq_E P_2 \implies GAADT(P_1) \subseteq_E GAADT(P_2))$ monotonicidade.

Lema 4.3.1. *O $GAADT$ é um processo evolutivo.*

Prova: Seja $P_t \in \mathbb{P}(P)$ logo, pela definição do $GAADT$ tem-se que:
Caso 1 - Se $P_t = P_{otm}$, então $GAADT(P_t) = P_{otm}$. Assim $P_t \equiv_E GAADT(P_t)$ e em conseqüência $P_t \subseteq_E GAADT(P_t)$. Logo $GAADT$ nesta situação é um processo evolutivo.
Caso 2a - Se $P_t \neq P_{otm}$ e $P_{otm} \subseteq_E P_t$, então $GAADT(P_t) = P_{otm}$ e temos pela definição de P_{otm}, que $adapt_m(P_{otm}) \geq k$, enquanto em P_t há pelo menos um elemento c cuja $adapt(c) < k$. Assim $adapt_m(P_t) \leq adapt_m(P_{otm})$ e deste modo $P_t \subseteq_E GAATD(P_t)$. Logo $GAADT$ neste caso é processo evolutivo.
Caso 2b - Se $P_t \neq P_{otm}$, e $P_{otm} \cap_E P_t \neq \emptyset$, então $GAADT(P_t) = P_{otm}$. Logo $GAADT$ nesta situação é um processo evolutivo.
Caso 2c - Se $P_t \neq P_{otm}$, e $P_{otm} \cap_E P_t = \emptyset$, então $GAADT(P_t) = P_{t+1}$, onde $P_{t+1} = cruz(c_1, c_2) \cup mut(c_3) \cup p_{corte}(P_t)$ e $p_{corte}(P_t) \neq \emptyset$, onde $c_1, c_2, c_3 \in P_t$. Vamos verificar se $adapt_m(P_t) \leq adapt_m(P_{t+1})$ e em conseqüência que $P_t \subseteq_E GAADT(P_t)$. De onde se conclui que $GAADT$ é um processo evolutivo.

Sendo $p_{corte} = adapt_m(P_t)$, então os cromossomos $c_1 \subseteq cruz(c_2, c_3)$ tem se que $adapt(c_1) \geq adapt_m(P_t)$, por estes cromossomos serem formados pelos genes de maior adaptação dos cromossomos da população P_t. Por outro lado para todo cromossomo $c_1 \subseteq mut(c_2)$, tem-se então que $adapt(c_1) \geq adapt(c_2)$, logo este cromossomo tem adaptação acima da média da adaptação dos cromossomos que não satisfazem o critério p_{corte}.

E finalmente qualquer cromossomo $c_1 \in p_{corte}(P_t)$ tem-se $adapt(c_1) \geq adapt_m(P_t)$. Assim conclui-se que $adapt_m(P_t) \leq adapt_m(P_{t+1})$, e em conseqüência que $P_t \subseteq_E GAADT(P_{t+1})$. As sucessivas aplicações do $GAADT$ sobre as populações P_t continuam até que uma das situações descritas acima se configurem.□

Lema 4.3.2. *O $GAADT$ é um processo evolutivo monotônico.*

Prova: Pelo Lema 4.3.1 o $GAADT$ é um processo evolutivo e deste modo resta mostrar que o $GAADT$ é monotônico. Com efeito pela Proposição 4.2.3 de sistema ordenado tem-se que $GAADT(P_1 \cap_E P_2) \subseteq_E GAADT(P_1 \cap_E P_2)$. Como por hipótese $P_1 \subseteq_E P_2$, então $P_1 \cap_E P_2 \equiv_E P_1$ e assim tem-se que $GAADT(P_1) \subseteq_E GAADT(P_1 \cap_E P_2)$. Por outro lado $GAADT(P_1 \cap_E P_2) \subseteq_E GAADT(P_2)$ porque pela Proposição 4.2.9 de interseção evolutiva tem-se que $adapt_m(P_1 \cap_E P_2) \leq adapt_m(P_2)$. Assim $GAADT(P_1) \subseteq_E GAADT(P_2)$, ou seja $\forall P_1, P_2 \in \mathbb{P}(P)(P_1 \subseteq_E P_2)$, então $GAADT(P_1) \subseteq_E GAADT(P_2)$ e portanto $GAADT$ é monotônico. □

Teorema 4.3.3. *(Convergência do $GAADT$) - O algoritmo genético $GAADT$ possui uma população estagnada, ou seja, é um processo evolutivo convergente.*

Prova: Lema 4.3.1 associado ao Lema 4.3.2. □

O significado do teorema acima é de grande importância para os estudo dos algoritmos genéticos e em particular para o estudo do $GAADT$, como está proposto em [43, 46, 44].

Devemos observar que a convergência do $GAADT$, estabelecida pelo teorema acima, não está subordinada a qualquer exigência especial. A sua convergência lhe é intrínseca, fazendo parte da sua definição. Os Lemas 4.3.1 e 4.3.2 são assertivas afirmativas singelas e decorre desta especificidade, o poder do Teorema 4.3.3 acima enunciado.

Capítulo 5

Aplicações do GAADT

5.1 Introdução

O GAADT foi proposto com o objetivo de ser aplicado a diferentes problemas, assim como o algoritmo genético de Holland e seus sucessores que foi adotado como método computacional para solucionar problemas de otimização de funções [54, 168, 92], otimização combinatória [46, 42], programação automática [105], aprendizagem [60, 165, 130, 152], economia [**?**], genética populacional [14], etc.. O sucesso da aplicação do GAADT no tratamento de um destes problemas depende de uma adequada interpretação das variáveis envolvidas na definição do problemas aos tipos básicos (base, gene, cromossomo e população) e aos símbolos relacionais, funcionais e constantes manipulados pelo algoritmo. Portanto, o GAADT, quando interpretado para um problema particular, deve preservar todas as suas propriedades fundamentais. Em outras palavras, o GAADT deve ser refinado pela sua versão interpretada, que, de agora em diante, será denominada genericamente de GAADTI.

A interpretação de um problema, através da especificação formal apresentada no Capítulo 3 para o GAADT, pode parecer uma tarefa árdua a princípio, pois o usuário terá que descrever de forma mais concreta os tipos, o estado do sistema, e os esquemas de operações, de modo que o GAADTI deve produzir um resultado válido, de acordo com o comportamento especificado para o GAADT. Uma proposta para diminuir a dificuldade inicial da adoção do GAADT, como uma ferramenta computacional para resolver problemas, é seguir uma metodologia que auxilie o usuário a modelar os requisitos do problema através dos tipos, das definições genéricas, das definições axiomáticas e dos esquemas usados para descrevê-lo. A metodologia usada nas aplicações do GAADT, apresentadas neste capítulo, compreende os seguintes passos:

1. Examinar o problema, com o objetivo de determinar qual a natureza da solução requerida no sentido de

orientar a escolha metafórica dos tipos básicos: base, gene e cromossomo;

2. Identificar o gene inócuo g_λ e a população inicial P_0;
3. Definir as funções grau de adaptação do gene $grau$ e peso do gene no cromossomo Θ, e a relação $atributoRelevante$;
4. Definir os elementos de Rq (F,M e r) usados pela função $cruz$ de acordo com o problema;
5. Definir os elementos de Rq (p_{corte}, t e k) utilizados pela função $GAADT$;
6. Construir o algoritmo;
7. Verificar os eventuais ajustes relativos à verificação dos tipos manipulados pelas funções, relações e operações manipuladas pela função $GAADT$, e a modelagem do problema através da definição dos componentes do ambiente.

Se o GAADT é refinado pelo GAADTI, então deve-se presumir que o GAADTI pode ser usado em qualquer lugar onde o GAADT for requerido. Isto acontece porque o estado e as operações do GAADTI refinam o estado e as operações do GAADT, de forma que o GAADTI refina o GAADT pela adição de algumas informações capazes de reduzir a indefinição e o indeterminismo presentes no estado e nas operações do GAADT.

Dados dois sistemas W e X especificados em Z, onde o sistema W é modelado em função do seu esquema de estado $StateW$, do seu esquema de inicialização $InitW$ e dos seus esquemas de operação $OpW_1, ..., OpW_j$; e o sistema X é modelado em função do seu esquema de estado $StateX$, do seu esquema de inicialização $InitX$ e dos seus esquemas de operação $OpX_1, ..., OpX_j$, diz-se que o sistema X refina o sistema W, se e somente se, existir uma relação $Retrieve$, que aqui é assumida ser funcional, entre o estado concreto $StateX$ e o estado abstrato $StateW$, tal que, considerando esta relação, as seguintes propriedades são satisfeitas:

- o esquema de inicialização $InitX$ é pelo menos tão determinístico quanto o esquema de inicialização $InitW$,
- para toda operação OpX_j a pré-condição desta operação é pelo menos tão fraca quanto a pré-condição da operação OpW_j correspondente, e
- para toda operação OpX_j o efeito desta operação é pelo menos tão determinístico quanto o efeito da operação OpW_j correspondente.

A pré-condição de uma operação $OpX_j = [\Delta StateX;\ i? : I;\ o! : O \mid PredicadoX_j]$ é um esquema que define a configuração do estado do sistema e das variáveis de entrada responsáveis por uma execução bem sucedida desta operação. Por isto, a parte declarativa do esquema de pré-condição da operação OpX_j é composta somente pelo estado do sistema antes da execução da operação ($StateX$) e pelas variáveis de entrada ($i?$), enquanto a parte predicativa diz que existe um estado após a execução da operação ($StateX'$) e variáveis de saída ($o!$), tal que $PredicadoX_{j_1}$ é satisfeito.

Definição 5.1.1. (Pré-condição) - Seja $OpX_j = [\Delta StateX;\ i? : I;\ o! : O \mid PredicadoX_j]$ uma operação, então $\mathsf{pre}\ OpX_j = [StateX;\ i? : I \mid \exists StateX';\ o! : O \bullet PredicadoX_j]$.

Definição 5.1.2. (Refinamento) - Sejam W e X especificações, em Z, abstrata e concreta, respectivamente, de um mesmo sistema, onde a especificação W é definida em função do seu esquema de estado $StateW$, do seu esquema de inicialização $InitW$ e dos seus esquemas de operação $OpW_1, ..., OpW_j$; e a especificação X é definida em função do seu esquema de estado $StateX$, do seu esquema de inicialização $InitX$ e dos seus esquemas de operação $OpX_1, ..., OpX_j$. Então, diz-se que o sistema X é um refinamento do sistema W, se as seguintes propriedades forem verdadeiras:

1. $\exists Retrieve : StateX \nrightarrow StateW \bullet true$,
2. $InitX \wedge Retrieve' \implies InitW$ (Teorema de Inicialização),
3. $\mathsf{pre}\ OpW_i \wedge Retrieve \implies \mathsf{pre}\ OpX_i$ (Teorema da Aplicabilidade),
4. $\mathsf{pre}\ OpW_i \wedge Retrieve \wedge OpX_i \wedge Retrieve' \implies OpW_i$ (Teorema da Corretude),

No GAADT, os elementos do problema são tratados como parâmetros das operações, que restringem a sua aplicação. Assim, dada uma instanciação do tipo base; das constantes; das funções $grau$, $peso$ e p_{corte}; e das relações M, F e $atributoRelevante$ do GAADT a um problema, obtém-se um GAADTI que preserva a definição do estado e das operações do algoritmo abstrato. A instanciação somente do tipo base se justifica pelo fato dos outros tipos manipulados pelo GAADT serem construídos a partir deste tipo, assim na realidade as transformações operadas pelo GAADT sobre os tipos gene e cromossomo podem ser redefinidas para trabalhar com os tipos seqüência base (gene) e conjunto de seqüência de base (cromossomo).

Desse modo, o GAADT pode ser visto como a função parametrizada pela seguinte estrutura $I_{GAADT} = \langle B, AFG, AFC, atributoRelevante, M, F, grau, \Theta, g_\lambda, p_{corte}, P_0, k, T\rangle$.

A Figura 5.1, ilustra como fica a definição da função $GAADT$, $GAADTI$ e $Retrieve$ considerando a ocorrência de uma instanciação e a parâmetrização das funções $GAADT$ e $GAADTI$, nesta figura a função $Instantiate$ definida na parte declarativa do esquema $Retrieve$ mostra a instanciação do conjunto de parâmetros na definição das operações GAADT e GAADTI.

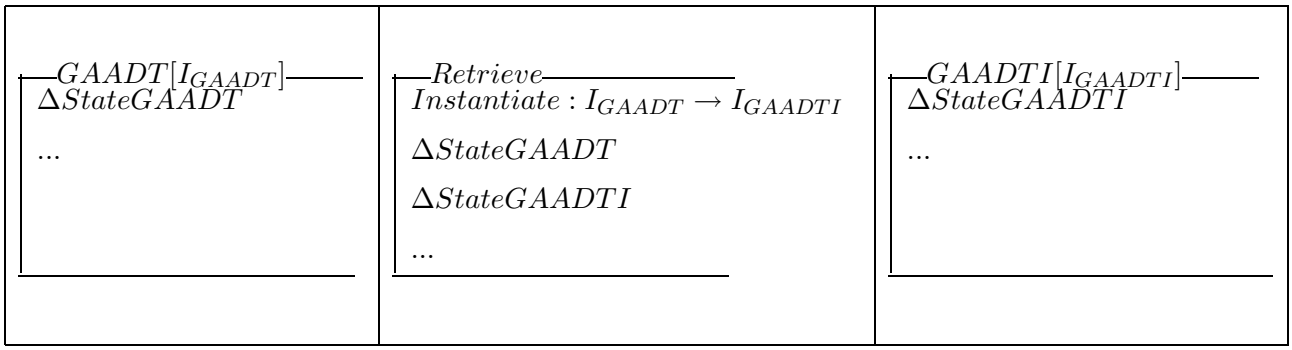


Figura 5.1: Esquema da interpretação do GAADT com Instanciação

Considerando que um problema é uma estrutura $Prob = \langle D, R, d, r, Adm\rangle$, onde D é o conjunto de dados, R é o conjunto de resultados, d é um predicado de seleção dos dados que interessam a $Prob$, r é a relação intencional (problema propriamente dito) entre D e R, e Adm é um a condição de admissibilidade de solução para o problema $Prob$, de acordo com [111, 2], então um problema $Prob$ genérico a ser resolvido pelo GAADT é uma estrutura do tipo $Prob = \langle \mathbb{P}(P), \mathbb{P}(P), \mathbb{P}(P), r, Alg\rangle$, onde: relação intencional do problema r é definida como $(P_1, P_1) \in r$, se e somente se, $adapt(P_1) \leq adapt(P_2)$ e $\forall P_3 \in \mathbb{P}(P)(adapt(P_3) \leq adapt(P_2))$ e Adm é a propriedade de ser algoritmo de busca. Dessa forma, a aplicação do GAADT para resolver um problema $Prob$ se restringe à instanciação factual da estrutura I_{GAADT} em combinação com a interpretação dada para os componentes da estrutura A que não pertencem a estrutura I_{GAADT}.

Em conseqüência disto, a função $Retrieve$ usada na prova de que GAADTI como um refinamento de GAADT é a função identidade, o que torna esta prova trivial, do ponto de vista de refinamento em Z. A única exigência que se faz de uma instanciação do GAADT é que os modelos usados para os *given sets* e as definições axiomáticas satisfaçam as propriedades originais. As vantagens advindas da preservação da estrutura do estado e das operações do GAADT em todas as suas instanciações é a garantia de satisfação das propriedades de corretude e aplicabilidade, e o aproveitamento do programa que implementa este algoritmo. Mas, mesmo que

a estrutura do estado e das operações entre o modelo abstrato e concreto não fossem preservadas o refinamento ainda estaria garantido se o estado e as operações concretas forem mas precisos do que os declarados no modelo abstrato. Mas, mesmo que a estrutura do estado e das operações entre o modelo abstrato e concreto não fossem preservadas, o refinamento ainda estaria garantido se o estado e as operações concretas forem mas precisos do que os declarados no modelo abstrato.

O restante deste capítulo se destina a descrever como o GAADT pode ser aplicado e refinado ao problema da construção da árvore filogenética de um conjunto de espécies a partir das informações contidas em uma matriz característica [46], ao problema de controle de semáforos de trânsito [45] e ao problema de monitoramento dos sinais vitais de pacientes na unidade de tratamento intensivo de um hospital [31]. Todos os dados apresentados neste capítulo foram gerados aleatoriamente para teste dos problemas acima referidos. As provas não apresentadas dos lemas, corolários e teoremas referenciados neste capítulo serão exibidas no Apêndice D.

5.2 Construção de Árvores Filogenéticas

O estudo da evolução das espécies é uma das áreas mais antigas e importantes das ciências biológicas e tem servido de fonte de informação a diversos campos da Biologia, tais como: Biogeografia, Biologia Molecular, Farmacologia, etc.

Na Biogeografia, as informações sobre a evolução das espécies permitem fazer uma reconstituição das regiões geográficas, que sofreram separação na época de sua formação. Esta reconstituição é feita através da análise da congruência entre padrões de distribuição de diferentes grupos de espécies [3, 107, 135].

Na Biologia Molecular, as informações sobre a evolução das espécies são utilizadas como um conhecimento adicional para auxiliar na decodificação do código genético (projeto GENOMA), ou melhor na busca para determinar a seqüência de bases nucleotídicas responsáveis por uma determinada característica [3, 30].

Na Farmacologia, as informações sobre a evolução das espécies têm direcionado os testes de reação imunológica, ou de toxidade no combate dos sintomas apresentados por uma família de vírus, de modo que estes testes tenham um alcance mais amplo e resultados mais previsíveis [3].

A confiança atribuída a uma dada informação sobre a evolução das espécies é diretamente proporcional ao volume de dados analisados na sua construção. Esta proporcionalidade fez com que a manipulação e o

tratamento das informações sobre a evolução das espécies fossem considerados tarefas difíceis e muito complexas para processamento manual. Por isto, as informações sobre a evolução das espécies passaram a ser representadas em diagramas ramificados, denominados árvores filogenéticas [15], e seu tratamento passou a ser executado com a ajuda de procedimentos automáticos [166].

Uma árvore filogenética $\mathbb{A}$ é uma árvore binária, cujos nós são ocupados pelas espécies $\mathbb{E}$ cuja história evolutiva está sendo investigada. Nesta árvore o arco que liga uma espécie pai a uma espécie filha é rotulado com o conjunto de características fenotípicas $\mathbb{F}$ que diferenciam as mesmas. As informações sobre as características presentes ou ausentes nas espécies investigadas são registradas em matrizes características $\mathbb{M}$. Esta matriz relaciona as espécies às características analisadas, tal que, se a característica j está presente na espécie e_w, com $1 \leq j \leq n_{\mathbb{F}}$ e $1 \leq w \leq n_{\mathbb{E}}$, então a célula $\mathbb{M}_{e_w,j} = 1$, senão $\mathbb{M}_{e_w,j} = 0$.

Definição 5.2.1. (Árvore Filogenética) - Uma árvore filogenética é uma estrutura $\mathbb{A}$ = ($\mathbb{E}$, $\mathbb{F}$, $éPai$, Γ), onde:

- $\mathbb{E}$ é o conjunto de $n_{\mathbb{E}}$ espécies,
- $\mathbb{F}$ é o conjunto de $n_{\mathbb{F}}$ fenótipos,
- $éPai$ é uma relação de ordem sobre os elementos de $\mathbb{E}$,
- $\Gamma : \mathbb{E} \rightarrow \mathbb{P}(F)$ é representado pelo grafo $\mathbb{G} = (\mathbb{E}, éPai)$ com cada arco $(e_{j,1}, e_{j,2})$ de $\mathbb{G}$ rotulado pela seqüência de elementos do conjunto $\Sigma_k = \Gamma(e_{j,2}) - \Gamma(e_{j,1})$.

Por exemplo, seja $\mathbb{A}$ = ($\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, $\{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$, $éPai$, Γ), onde $\{(e_1,e_2), (e_1,e_3), (e_3,e_4), (e_3,e_5), (e_5,e_6), (e_5,e_7)\} \subseteq éPai$, e $\{(e_1,\{c_1, c_2, c_3\}), (e_2,\{c_4, c_2, c_3\}), (e_3,\{c_1, c_5, c_3\}), (e_4,\{c_4, c_6, c_3\}), (e_5,\{c_1, c_5, c_7\}), (e_6,\{c_4, c_5, c_7\}), (e_7,\{c_1, c_5, c_7\})\ \} \subseteq \Gamma$. A Figura 5.2 mostra a árvore filogenética e a Tabela 5.1 mostra a matriz característica para esta estrutura $\mathbb{A}$.

Nesta seção, o GAADT será a ferramenta computacional usada para construir a árvore filogenética de um conjunto de espécies a partir dos dados armazenados na matriz característica das espécies em análise.

5.2.1 Tipos Básicos

O tipo base para a construção da árvore filogenética a partir das informações contidas na matriz característica da Tabela 5.1 é o conjunto formado pelo nome das espécies investigadas acrescido de uma espécie denominada de inócua (e_λ), ou seja, $BI ::= e_1 \mid e_2 \mid e_3 \mid e_4 \mid e_5 \mid e_6 \mid e_7 \mid e_\lambda$.

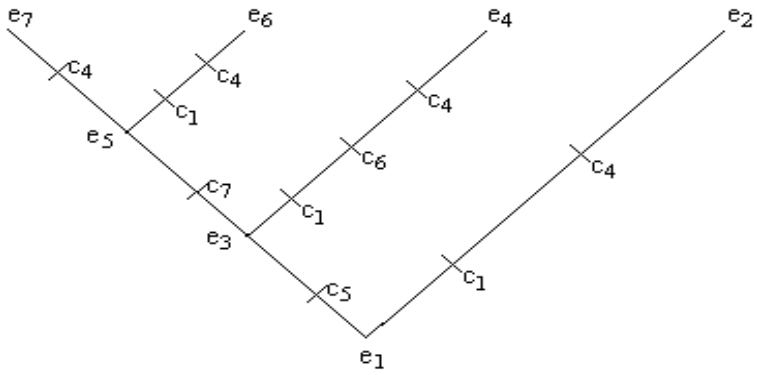


Figura 5.2: Árvore filogenética

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
e_1	1	1	1	0	0	0	0
e_2	0	1	1	1	0	0	0
e_3	1	0	1	0	1	0	0
e_4	0	0	1	1	0	1	0
e_5	1	0	0	0	1	0	1
e_6	0	0	0	1	1	0	1
e_7	1	0	0	1	0	0	1

Tabela 5.1: Matriz característica

A interpretação dada para o tipo gene $g = \langle e_y, e_x, e_z \rangle$ é que ele representa uma árvore elementar, com raiz e_x, a folha à esquerda é e_y e a folha à direita é e_z, a qual satisfaz o seguinte conjunto de axiomas de formação de genes:

- para todo gene, se a raiz for igual à base inócua, então as folhas também devem ser iguais à base inócua;
- para todo gene, se a raiz é diferente da base inócua, então as folhas devem ser diferentes entre si;
- para todo gene, se tanto a raiz como as folhas são diferentes da base inócua, então a raiz e as folhas são diferentes entre si;
- para todo gene, os pares formados pela raiz e cada uma das folhas pertencem à relação $\acute{e}Pai$.

O acesso ao elemento da base que representa a raiz, a folha à direita ou à esquerda de um gene será feito através das funções $raiz$, $direita$ e $esquerda$, definidas como:

$$
\begin{array}{|l}
raiz : (\ \mathsf{seq}\ BI) \pfun BI \\
direita : (\ \mathsf{seq}\ BI) \pfun BI \\
esquerda : (\ \mathsf{seq}\ BI) \pfun BI \\
\hline
[RaizDef] \\
\forall g :\ \mathsf{seq}\ BI \mid \#g = 3 \bullet raiz(g) = head\ (tail\ (g)) \\
[DireitaDef] \\
\forall g :\ \mathsf{seq}\ BI \mid \#g = 3 \bullet direita(g) = last\ (g) \\
[EsquerdaDef] \\
\forall g :\ \mathsf{seq}\ BI \mid \#g = 3 \bullet esquerda(g) = head\ (g)
\end{array}
$$

Antes de se definir o tipo gene, é necessário apresentar a especificação da relação $éPai$, utilizada na descrição do conjunto de axiomas de formação de genes.

$$
\begin{array}{|l}
éPai : BI \leftrightarrow BI \\
\hline
[PaiDef] \\
éPai = \{(e_1, e_2), (e_1, e_3), (e_3, e_4), (e_3, e_5), (e_5, e_6), (e_5, e_7), (e_\lambda, e_\lambda)\}
\end{array}
$$

Note que nesta definição, além das informações declaradas para a relação $éPai$, na Seção 5.2, um elemento a mais foi incluído para informar que a espécie inócua só se relaciona com ela própria.

O conjunto de axiomas de formação de genes para este exemplo é:

$$
\begin{array}{|l}
AFGI : \mathbb{P}(\ \mathsf{seq}\ BI) \\
\hline
(\forall g : AFGI \bullet \#g = 3) \land \\
(\forall g : AFGI \mid raiz(g) = e_\lambda \bullet direita(g) = e_\lambda \land esquerda(g) = e_\lambda) \\
(\forall g : AFGI \mid raiz(g) \neq e_\lambda \bullet direita(g) \neq raiz(g) \land esquerda(g) \neq raiz(g)) \\
(\forall g : AFGI \mid e_\lambda \notin\ \mathsf{ran}\ g \bullet raiz(g) \neq direita(g) \land raiz(g) \neq esquerda(g) \land direita(g) \neq esquerda(g)) \\
(\forall g : AFGI \bullet (raiz(g), direita(g)) \in éPai \land (raiz(g), esquerda(g)) \in éPai)
\end{array}
$$

Agora o tipo gene pode ser definido como $GI == AFGI$, lê-se o conjunto das seqüências cujo alfabeto é o conjunto BI, que obedece as restrições imposta pelos $AFGI$.

O gene inócuo neste caso é a constante $\langle e_\lambda,e_\lambda,e_\lambda \rangle$ do sistema.

$$\frac{gI_\lambda : GI}{gI_\lambda = \langle e_\lambda, e_\lambda, e_\lambda \rangle}$$

O cromossomo é interpretado como a árvore filogenética construída pela sobreposição das árvores elementares que o compõem. Os axiomas de formação de cromossomos, neste caso, exigem que:

- todas as espécies do conjunto base estejam presentes no cromossomo;
- o número máximo de ocorrências de uma base como raiz no cromossomo é 1;
- o número máximo de ocorrências de uma base como folha no cromossomo é menor ou igual à 1;
- o número de bases cuja sua única ocorrência no cromossomo é como raiz, é 1.

Assim, o conjunto de axiomas de formação de cromossomos é especificado como:

$$\frac{AFCI : \mathbb{P}(\mathbb{P}(GI))}{\begin{array}{l}(\forall c : AFCI \bullet \{b : BI \mid \exists g : c \bullet b \in \mathsf{ran}\ g\} = BI) \land \\ (\forall c : AFCI \bullet \forall b : BI \bullet \#\{g : c \mid raiz(g) = b\} = 1) \land \\ (\forall c : AFCI \bullet \forall b : BI \bullet \#\{g : c \mid direita(g) = b\} + \#\{g : c \mid esquerda(g) = b\} \leq 1) \land \\ (\forall c : AFCI \bullet \#(\{b : BI \mid \forall g : c \bullet raiz(g) = b\} \\ \setminus \{b : BI \mid \forall g : c \bullet direita(g) = b \lor esquerda(g) = b\}) = 1)\end{array}}$$

O tipo cromossomo fica especificado como $CI == AFCI$, lê-se o conjunto de genes que atendem as exigências dos AFCI, enquanto o tipo população é um conjunto de conjuntos de cromossomos, $PI == \mathbb{P}(\mathbb{P}(CI))$.

Para este exemplo será assumido que a população inicial é formada pelo cromossomo que representa a árvore filogenética da Figura 5.2, ou seja,

$$PI_0 : PI$$
$$PI_0 = \{\{(e_2, e_1, e_3), (e_4, e_3, e_5), (e_6, e_5, e_7)\}\}.$$

5.2.2 Operadores Genéticos

Assim como a definição do estado e das operações, existem algumas definições axiomáticas do GAADT que também são estruturalmente preservadas por instanciação. Por exemplo, a Tabela 5.2 ilustra algumas definições cujas estruturas são trivialmente preservadas. A demais definições são preservadas de forma análoga.

$\equiv_C : C \leftrightarrow C$ $[eqDef]$ $\forall c_1, c_2 : C \bullet$ $c_1 \setminus \{g_\lambda\} = c_2 \setminus \{g_\lambda\} \Leftrightarrow (c_1, c_2) \in \equiv_C$	$\equiv I_C : CI \leftrightarrow CI$ $[eqIDef]$ $\forall c_1, c_2 : CI \bullet$ $c_1 \setminus \{gI_\lambda\} = c_2 \setminus \{gI_\lambda\} \Leftrightarrow (c_1, c_2) \in \equiv I_C$
$mesma : G \leftrightarrow G$ $[mesmaDef]$ $\forall g_1, g_2 : G \bullet (g_1, g_2) \in mesma$ $\Leftrightarrow (\exists G_1 : atributoRelevante \bullet \{g_1, g_2\} \subseteq G_1)$	$mesmaI : GI \leftrightarrow GI$ $[mesmaIDef]$ $\forall g_1, g_2 : GI \bullet (g_1, g_2) \in mesmaI$ $\Leftrightarrow (\exists G_1 : atributoRelevanteI \bullet \{g_1, g_2\} \subseteq G_1)$

Tabela 5.2: Definições axiomáticas preservadas

O grau de adaptação dos genes para a construção de árvores filogenéticas é calculado pela seguinte fórmula:

$$grauI(\langle e_y, e_x, e_z \rangle) = \begin{cases} 0 & \text{se } e_y = e_x = e_z = e_\lambda, \\ homologia(\langle e_y, e_x, e_z \rangle) & \text{caso contrário} \end{cases}$$

onde $homologia(\langle e_y, e_x, e_z\rangle) = \sum_{i=1}^{n_{\mathbb{F}}} \mid \mathbb{M}_{e_y,i} - \mathbb{M}_{e_x,i} \mid + \mid \mathbb{M}_{e_z,i} - \mathbb{M}_{e_x,i} \mid$.

As homologias existentes entre a raiz e as folhas da árvore filogenética simples, representada pelos elementos pertencentes ao tipo GI, com base nas informações contidas na Tabela 5.1, são especificadas pela seguinte definição axiomática:

$$
\begin{array}{|l}
homologia : BI \times BI \pfun \mathbb{Q} \\
\hline
[HomologiaInocuoDef] \\
\forall b_1, b_2 : BI \bullet b_2 = e_\lambda \Leftrightarrow homologia(b_1, b_2) = (4, 1) \\
[HomologiaMesmoDef] \\
\forall b_1, b_2 : BI \bullet b_1 = b_2 \Leftrightarrow homologia(b_1, b_2) = (4, 1) \\
[HomologiaDef] \\
homologia(e_1, e_2) = (5, 1) \land homologia(e_1, e_3) = (5, 1) \land homologia(e_1, e_4) = (3, 1) \land \\
homologia(e_1, e_5) = (3, 1) \land homologia(e_1, e_6) = (1, 1) \land homologia(e_1, e_7) = (3, 1) \land \\
homologia(e_2, e_3) = (3, 1) \land homologia(e_2, e_4) = (5, 1) \land homologia(e_2, e_5) = (1, 1) \land \\
homologia(e_2, e_6) = (3, 1) \land homologia(e_2, e_7) = (3, 1) \land homologia(e_3, e_4) = (3, 1) \land \\
homologia(e_3, e_5) = (5, 1) \land homologia(e_3, e_6) = (3, 1) \land homologia(e_3, e_7) = (3, 1) \land \\
homologia(e_4, e_5) = (1, 1) \land homologia(e_4, e_6) = (3, 1) \land homologia(e_4, e_7) = (3, 1) \land \\
homologia(e_5, e_6) = (5, 1) \land homologia(e_5, e_7) = (5, 1) \land homologia(e_6, e_7) = (5, 1) \\
[HomologiaComutativaDef] \\
\forall b_1, b_2 : BI \bullet homologia(b_1, b_2) = homologia(b_2, b_1)
\end{array}
$$

Lema 5.2.2. $\forall b_1, b_2 : BI \bullet (homologia(b_1, b_2), (0, 1)) \in maior\mathbb{Q}$

A função $grauI$ mapeia o gene inócuo no número racional (0,1), e todos os outros genes no número racional resultante da soma racional das homologias existentes entre as folhas e a raiz deste gene.

$$
\begin{array}{|l}
grauI : GI \pfun \mathbb{Q} \\
\hline
[grauIInocuoProp1Def] \\
grauI(gI_\lambda) = (0, 1) \\
[GrauINaoInocuoDef] \\
\forall g : GI \mid g \neq gI_\lambda \bullet \\
grauI(g) = adicao\mathbb{Q}(homologia(direita(g), raiz(g)), homologia(esquerda(g), raiz(g)))
\end{array}
$$

Lema 5.2.3. $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), grauI(gI_\lambda)) \in maior\mathbb{Q}$.

O atributo relevante para a construção de árvores filogenéticas são as raízes das árvores elementares que a compõem. Isto porque está implícito no processo de construção destas árvores que o seu objetivo é encontrar o conjunto de árvores elementares cujo número de características compartilhadas entre a espécie raiz e as espécies folhas é máximo.

$$
\begin{array}{|l}
atributoRelevanteI : \mathbb{P}(\mathbb{P}(GI)) \\
\hline
[atributoRelevanteIDef] \\
\forall g_1, g_2 : GI \bullet raiz(g_1) = raiz(g_2) \Leftrightarrow (\exists G_1 : atributoRelevanteI \bullet \{g_1, g_2\} \subseteq G_1)
\end{array}
$$

Lema 5.2.4. $\forall g : GI \bullet (g, g) \in mesmaI$

Lema 5.2.5. $\forall g_1, g_2 : GI \mid (g_1, g_2) \in mesmaI \bullet (g_2, g_1) \in mesmaI$

Lema 5.2.6. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$

Para verificar se a definição da relação mesma característica e da função grau de adaptação do gene estão corretas com o modelo do GAADT, basta provar que ela ainda atende as propriedades reflexiva e associativa exigidas na sua definição.

Lema 5.2.7. $\forall g : GI \mid g = gI_\lambda \bullet domiI(g, g) = g \Leftrightarrow (g, g) \in mesmaI$

Lema 5.2.8. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_1, g_3) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet$
$domiI(g_1, domiI(g_2, g_3)) = domiI(domiI(g_1, g_2), g_3)$

O peso de um gene g no cromossomo c para este caso é (1,1) se o gene g pertencer ao conjunto de genes que compõe o cromossomo c, caso contrário ele é (0,1).

$$\Theta I : CI \times GI \nrightarrow \mathbb{Q}^+$$

$$[ThetaINaoPertenceDef]$$
$$\forall c : CI;\ g : GI \mid g \notin c \bullet \Theta I(c, g) = (0, 1)$$
$$[ThetaIPertenceDef]$$
$$\forall c : CI;\ g : GI \mid g \in c \bullet \Theta I(c, g) = (1, 1)$$

O predicado rI do ambiente usado pela função de seleção, para definir a população de cromossomos que podem ser considerados habilitados para cruzamento, no caso do problema considerado nesta seção não apresenta nenhuma exigê, ficando assim igual ao tipo população, formada por todos os cromossomos do tipo CI.

$$rI : \mathbb{P}(PI)$$

$$rI = CI$$

De posse desta definição axiomática usada pela função $selI$, que por sua vez é usada pela função $fecI$, pode-se provar que a função $fecI$ é reflexiva.

Lema 5.2.9. $\forall c : CI \bullet (fecI(c, c), c) \in\equiv I_C$

O comportamento da função $cruzI$ para o problema de construção de árvores filogenéticas dependerá da função $fecI$, que mantém seu modelo original, e da definição dos predicados MI e FI, que neste caso aceitará todos os cromossomos pertencentes ao conjunto $AFCI$.

$$MI : \mathbb{P}(PI)$$
$$FI : \mathbb{P}(PI)$$

$$MI = PI \wedge$$
$$FI = PI$$

5.2.3 Algoritmo

Antes de apresentar os esquemas do GAADT instanciado para o problema de construção de árvores filogenéticas é preciso exibir a função ponto de corte e adaptação média, e o parâmetro de satisfação da questão. A adaptação média, por ser uma medida, não terá sua definição alterada; ela será simplesmente instanciada para os tipos declarados na Seção 5.2.1.

$$adaptTotalI : PI \pfun \mathbb{Q}^+$$

$$[adaptTotalIDef]$$
$$adaptTotalI(\emptyset) = (0,1) \land$$
$$(\forall PI_1 : PI \bullet \forall c : PI_1 \bullet adaptTotalI(PI_1) = adicao\mathbb{Q}(adaptI(c), adaptTotalI(PI_1 \setminus \{c\})))$$

$$adaptMediaI : PI \pfun \mathbb{Q}^+$$

$$[adaptMediaIDef]$$
$$adaptMediaI(\emptyset) = (0,1) \land$$
$$(\forall PI_1 : PI \bullet adaptMediaI(PI_1) = divisao\mathbb{Q}(adaptTotalI(PI_1), (\#PI_1, 1)))$$

O ponto de corte adotado para este problema será o conjunto dos cromossomos com adaptação abaixo da média.

$$pI_{corte} : PI \pfun PI$$

$$\forall PI_1 : PI \bullet pI_{corte}(PI_1) = \{c : PI_1 \mid maior\mathbb{Q}(adaptI(c), adaptMediaI(PI_1)) \lor$$
$$mesmo\mathbb{Q}(adaptI(c), adaptMediaI(PI_1))\}$$

Os valores adotados para as duas primeiras alternativas de parada utilizadas na definição da função *GAADTI*, e referidas como número máximo de iterações e adaptação do cromossomo considerada satisfatória para o problema em foco, são definidos abaixo.

$$
\begin{array}{|l}
TI : \mathbb{Z} \\
kI : \mathbb{Q} \\
\hline
TI = 100 \land kI = (0, 1)
\end{array}
$$

Neste caso, como no exemplo do problema do caixeiro viajante, o algoritmo só irá parar quando atingir a iteração 100, já que é impossível gerar um cromossomo com grau de adaptação igual a zero segundo a definição apresentada para o tipo CI. Com base nas definições axiomáticas descritas na Seção anterior, tem-se que o modelo do GAADT quando instanciado para o problema de construção de árvores filogenéticas mantém a definição do seu estado e das suas operações, como ilustrado na Tabela 5.3. As demais operações preservam a estrutura de forma análoga.

$StateGAADT$ $Tx : C \leftrightarrow C$ $P_{atual} : P$ $P_{cruz} : P$ $P_{mut} : P$ $t : \mathbb{Z}$ $MACHO : P$ $FEMEA : P$ --- $MACHO \subseteq P_{atual} \land FEMEA \subseteq P_{atual} \land$ $(\forall c_1, c_2 : C \mid (c_1, c_2) \in Tx \bullet$ $(\forall c_3 : P_{atual} \bullet (c_1, c_3) \notin \equiv_C))$	$StateGAADTI$ $TxI : CI \leftrightarrow CI$ $PI_{atual} : PI$ $PI_{cruz} : PI$ $PI_{mut} : PI$ $tI : \mathbb{Z}$ $MACHOI : PI$ $FEMEAI : PI$ --- $MACHOI \subseteq PI_{atual} \land FEMEAI \subseteq PI_{atual} \land$ $(\forall c_1, c_2 : CI \mid (c_1, c_2) \in TxI \bullet$ $(\forall c_3 : PI_{atual} \bullet (c_1, c_3) \notin \equiv I_C))$
$InitGAADT$ $StateGAADT'$ --- $Tx' = \emptyset \land$ $p'_{atual} = P_0 \land p'_{cruz} = \emptyset \land p'_{mut} = \emptyset \land$	$InitGAADTI$ $StateGAADTI'$ --- $TxI' = \emptyset \land$ $pI'_{atual} = PI_0 \land pI'_{cruz} = \emptyset \land pI'_{mut} = \emptyset \land$

$t' = \emptyset \wedge$ $MACHO' = sel(p_{corte}(P_0), M) \wedge$ $FEMEA' = sel(p_{corte}(P_0), F)$	$tI' = \emptyset \wedge$ $MACHOI' = selI(pI_{corte}(PI_0), MI) \wedge$ $FEMEAI' = selI(pI_{corte}(PI_0), FI)$
Cruzamento $\Delta StateGAADT$ $Tx' = Tx \wedge P'_{atual} = P_{atual} \wedge$ $P'_{cruz} = \{c_1 : C \mid (\exists c_2 : MACHO;$ $c_3 : FEMEA \bullet c_1 \in cruz(c_2, c_3)) \wedge$ $(\forall c_4, c_5 : C \mid (c_4, c_5) \in Tx \bullet$ $(c_4, c_1) \notin \equiv_C \wedge (c_5, c_1) \notin \equiv_C)\} \wedge$ $P'_{mut} = P_{mut} \wedge t' = t \wedge$ $MACHO' = MACHO \wedge$ $FEMEA' = FEMEA$	*CruzamentoI* $\Delta StateGAADTI$ $TxI' = TxI \wedge PI'_{atual} = PI_{atual} \wedge$ $PI'_{cruz} = \{c_1 : CI \mid (\exists c_2 : MACHOI;$ $c_3 : FEMEAI \bullet c_1 \in cruzI(c_2, c_3)) \wedge$ $(\forall c_4, c_5 : CI \mid (c_4, c_5) \in TxI \bullet$ $(c_4, c_1) \notin \equiv I_C \wedge (c_5, c_1) \notin \equiv I_C)\} \wedge$ $PI'_{mut} = PI_{mut} \wedge tI' = tI \wedge$ $MACHOI' = MACHOI \wedge$ $FEMEAI' = FEMEAI$

Tabela 5.3: Definições de operações preservadas

Apesar do refinamento do GAADT ser trivial pela preservação da estrutura do seu estado, dos seus esquemas de operações e das definições axiomáticas, como pode-se verificar pela Tabela 5.3, a caracterização geral do processo de refinamento será apresentada na Seção 5.5.

5.3 Controle de Semáforos de Trânsito

O problema de controle de semáforos de trânsito das cidades se resume em determinar o tempo em que os semáforos devem ou não ficar no estado vermelho, ou verde, ou amarelo para evitar congestionamentos. Esses congestionamentos podem ser causados por situações imprevisíveis tais como: um acidente de trânsito, um evento (a inauguração de uma loja), uma passeata política, etc, ou por situações previsíveis como: os horários de pico nas ruas que dão acesso as regiões industriais, comerciais, escolares de uma cidade, ou o desvio do trânsito de uma região onde está ocorrendo uma festa popular, ou um conserto de rua. Em ambos os casos,

serão necessários alterar os tempos em que os semáforos da região congestionada e das regiões adjacentes, ficam nos estados vermelho, verde e amarelo de modo a impedir que o congestionamento se perpetue por muito tempo.

No caso das situações previsíveis, não é tão difícil evitar o congestionamento, já que a causa do congestionamento é conhecida a priori. Mas diante de situações imprevisíveis, não existe forma de se ter um planejamento, uma vez que as causas podem ser de uma variedade muito grande e consequentemente podem gerar um número de alternativas explosivo. O tratamento nesses casos deve levar em conta a situação factual *in loco* diante das efemérides que estejam ocorrendo. O sistema de controle de semáforos de trânsito de veículos nas cidades, proposto nesse trabalho, é especialmente recomendado para evitar congestionamentos provocados por situações imprevisíveis. Seu objetivo é alterar os tempos em que os semáforos da região e das regiões adjacentes atingidas, por um congestionamento, ficam nos estados vermelho, verde e amarelo. Para tanto, esse sistema recebe informações em tempo real do fluxo de veículos em todos os pontos da cidade, com semáforos provido de sensores a eles adaptados. Essas informações servem para o sistema identificar as regiões onde estão ocorrendo fluxos normais ou anormais.

O controle de semáforo de trânsito nas cidades é um problema de sistema adaptativo complexo por natureza, pois os tempos em que os semáforos de uma região devem ficar nos estados vermelho, verde e amarelo, para manter o fluxo de veículos normal, dependem das interações não-lineares de vários agentes adaptativos, tais como: dos motoristas de veículos, dos pedestres, das dimensões das ruas, dos acontecimentos sociais fortúitos, do clima, do horário, da região, dos consertos entre muitíssimos outros fatores.

Nesta seção, uma instanciação do GAADT será usada para controlar os semáforos de trânsito de uma cidade. Nesta instanciação só serão apresentadas as especificações dos tipos, dos conjuntos, das relações, das funções e das operações que forem diferentes das apresentadas nas Tabelas 5.2 e 5.3.

5.3.1 Tipos Básicos

O tipo base para o sistema de controle de semáforos é interpretado como sendo o conjunto de funções que mapeiam cada semáforo a uma seqüência de números naturais $\langle verde, amarelo, vermelho \rangle$, os termos desta seqüência indicam o tempo que o semáforo permanece nas cores verde, amarelo e vermelho respectivamente. Entre os semáforos trabalhados pelo sistema existe um que não faz parte da rede viária da cidade e que é usado na construção do gene inócuo, tal semáforo é denominado de semáforo inócuo e representado pelo símbolo

$semaforo_\lambda$.

$$[SEMAFORO]$$
$$BASEI == SEMAFORO \pfun \mathsf{seq}\ \mathbb{N}$$

O tipo gene apresenta a seguinte composição $g = \langle semaforo \pfun \langle verde, amarelo, vermelho \rangle \rangle$, o qual satisfaz o seguinte conjunto de axiomas de formação de genes:

- o parâmetro $verde$ de cada semáforo deve pertencer a um intervalo limitado pelas constantes *VerdeMaximo* e *VerdeMinimo*;
- o parâmetro $amarelo$ de cada semáforo deve pertencer a um intervalo limitado pelas constantes *AmareloMaximo* e *AmareloMinimo*; e
- o parâmetro $vermelho$ de cada semáforo deve pertencer a um intervalo limitado pelas constantes *VermelhoMaximo* e *VermelhoMinimo*.

$$VerdeMaximo : SEMAFORO \pfun \mathbb{N}$$
$$VerdeMinimo : SEMAFORO \pfun \mathbb{N}$$
$$AmareloMaximo : SEMAFORO \pfun \mathbb{N}$$
$$AmareloMinimo : SEMAFORO \pfun \mathbb{N}$$
$$VermelhoMaximo : SEMAFORO \pfun \mathbb{N}$$
$$VermelhoMinimo : SEMAFORO \pfun \mathbb{N}$$

$$[VerdeInteraloDef]$$
$$(\forall s : SEMAFORO \bullet VerdeMaximo(s) > 0 \land VerdeMaximo(s) \geq VerdeMinimo(s))$$
$$[AmareloInteraloDef]$$
$$(\forall s : SEMAFORO \bullet AmareloMaximo(s) > 0 \land AmareloMaximo(s) \geq AmareloMinimo(s))$$
$$[VermelhoInteraloDef]$$
$$(\forall s : SEMAFORO \bullet VermelhoMaximo(s) > 0 \land VermelhoMaximo(s) \geq VermelhoMinimo(s))$$

$$
\begin{array}{|l}
afgI : \mathbb{P}(\ \mathsf{seq}\ BASE) \\
\hline
\forall g : afgI \bullet \#g = 1 \land \\
\quad (\exists s : SEMAFORO \bullet \\
\qquad (\exists b : BASE;\ n : \mathbb{N} \bullet b = head\ (g) \land s \in\ \mathsf{dom}\ b \land \\
\qquad\quad n = head\ (b(s)) \land n \leq VerdeMaximo(s) \land n \geq VerdeMinimo(s)) \land \\
\qquad (\exists b : BASE;\ n : \mathbb{N} \bullet b = head\ (tail\ (g)) \land s \in\ \mathsf{dom}\ b \land \\
\qquad\quad n = head\ (tail\ (b(s))) \land n \leq AmareloMaximo(s) \land n \geq AmareloMinimo(s)) \land \\
\qquad (\exists b : BASE;\ n : \mathbb{N} \bullet b = head\ (tail\ (tail\ (g))) \land s \in\ \mathsf{dom}\ b \land \\
\qquad\quad n = head\ (tail\ (tail\ (b(s)))) \land n \leq VerdeMaximo(s) \land n \geq VerdeMinimo(s)))
\end{array}
$$

O gene inócuo neste caso é representado pela função que mapeia o semáforo inócuo a qualquer valor de duração de tempo das cores verde, amarelo e vermelho pertencente ao intervalo de tempo previamente estipulado para as mesmas.

$$
\begin{array}{|l}
semaforo_\lambda : SEMAFORO \\
gI_\lambda :\ \mathsf{seq}\ BASE \\
\hline
\#gI_\lambda = 1 \land \\
(\forall b : BASE \bullet b = head\ (gI_\lambda) \land semaforo_\lambda \in\ \mathsf{dom}\ b)
\end{array}
$$

O tipo cromossomo é interpretado como sendo uma região da cidade que contém o semáforo com congestionamento, o qual satisfaz o seguinte conjunto de axiomas de formação de cromossomo:

- não existem dois genes no cromossomo que representem um mesmo semáforo; e
- existe um gene no cromossomo que representa o semáforo onde o congestionamente no trânsito da cidade está ocorrendo, todo os outros genes no cromossomo são vizinhos deste semáforo, a noção de vizinhança faz parte do projeto de trânsito das cidades e compreende o conjunto de semáforos que liberam, recebem ou retem o fluxo de veículos do semáforo.

$$
\begin{array}{|l}
Vizinhanca : SEMAFORO \pfun \mathbb{P} SEMAFORO \\
afcI : \mathbb{P}(\mathbb{P} GENEI) \\
\hline
\forall c : afcI \bullet \\
\quad (\exists s_1, s_2 : SEMAFORO;\ g_1, g_2 : GENEI;\ b_1, b_2 : BASE \bullet \{g_1, g_2\} \subseteq c \land g_1 \neq g_2 \land \\
\qquad b_1 = head\,(g_1) \land b_2 = head\,(g_2) \land s_1 \in \mathsf{dom}\ b_1 \land s_2 \in \mathsf{dom}\ b_2 \land s_1 \neq s_2) \land \\
\quad (\exists s : SEMAFORO;\ g_1 : GENEI;\ b_1 : BASE \bullet g_1 \in c \land b_1 = head\,(g_1) \land s \in \mathsf{dom}\ b_1 \land \\
\qquad (\forall g_2 : GENEI;\ b_1 : BASE \bullet g_2 \in c \setminus \{g_1\} \land b_2 = head\,(g_2) \land \mathsf{dom}\ b_2 \subseteq Vizinhanca(s)))
\end{array}
$$

5.3.2 Operadores Genéticos

O grau de adaptação de um gene $g = \langle semaforo \pfun \langle verde, amarelo, vermelho \rangle \rangle$ para o sistema de controle de semáforo é zero, um e doi dependendo do valor da diferença entre o fluxo lido e o fluxo padrão estabelecido no projeto viário da cidade para cada semáforo. Quando o fluxo no semáforo representado pelo gene está abaixo do esperado seu grau de adaptação é um, igual ao esperado seu grau é zero e acima do esperado seu grau é dois. É justamente sobre os semáforos representados pelos genes com grau de adaptação dois que o

algoritmo genético especificado nesta seção irá trabalhar. O grau de adaptação do gene inócuo é zero.

$$
\begin{array}{|l}
\\
FluxoPadrao : SEMAFORO \pfun \nat \\
FluxoLido : SEMAFORO \pfun \nat \\
grauI : GI \pfun \rat \\
\hline
[grauIInocuoProp1Def] \\
grauI(gI_\lambda) = (0, 1) \\
[GrauIUmDef] \\
\forall g : GI \mid g \neq gI_\lambda \bullet \\
\quad (\exists s : SEMAFORO \bullet \\
\qquad (\exists b : BASE;\ n : \mathbb{Z} \mid n \geq 0 \bullet b = head\,(g) \land s \in \mathsf{dom}\ b \land \\
\qquad\quad n = FluxoPadrao(s) - FluxoLido(s) \land grauI(g) = (1, 1))) \\
[GrauIDoisDef] \\
\forall g : GI \mid g \neq gI_\lambda \bullet \\
\quad (\exists s : SEMAFORO \bullet \\
\qquad (\exists b : BASE;\ n : \mathbb{Z} \mid n < 0 \bullet b = head\,(g) \land s \in \mathsf{dom}\ b \land \\
\qquad\quad n = FluxoPadrao(s) - FluxoLido(s) \land grauI(g) = (2, 1)))
\end{array}
$$

Lema 5.3.1. $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), grauI(gI_\lambda)) \in maior\mathbb{Q}$.

O atributo relevante para o gene é o parâmetro $semaforo$ relacionado ao mesmo. Isto porque a manutenção de um fluxo de trânsito normal na cidade requer que cada semáforo da cidade esteja configurado com durações de tempos apropriados para as cores verde, amarelo e vermelho.

$$
\begin{array}{|l}
\\
atributoRelevanteI : \pow(\pow(GI)) \\
\hline
[atributoRelevanteIDef] \\
\forall g_1, g_2 : GI \bullet \\
\quad (\exists s_1, s_2 : SEMAFORO \bullet \\
\qquad (\exists b_1, b_2 : BASE \bullet b_1 = head\,(g_1) \land b_2 = head\,(g_2) \land s_1 \in \mathsf{dom}\ b_1 \land s_2 \in \mathsf{dom}\ b_2 \land s_1 = s_2))
\end{array}
$$

As definições da relação mesma característica e da função grau de adaptação do gene interpretadas para o sistema de controle de semáforo estão corretas com o modelo do GAADT se elas satisfizerem as propriedades reflexiva e associativa.

Lema 5.3.2. $\forall g : GI \bullet (g, g) \in mesmaI$

Lema 5.3.3. $\forall g_1, g_2 : GI \mid (g_1, g_2) \in mesmaI \bullet (g_2, g_1) \in mesmaI$

Lema 5.3.4. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$

Lema 5.3.5. $\forall g : GI \mid g = gI_\lambda \bullet domiI(g, g) = g \Leftrightarrow (g, g) \in mesmaI$

Lema 5.3.6. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_1, g_3) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet$
$domiI(g_1, domiI(g_2, g_3)) = domiI(domiI(g_1, g_2), g_3)$

O peso de um gene g no cromossomo c para o problema de controle de semáforo é igual a $(0, 1)$ se o grau de adaptação deste gene menor ou igual à $(1, 1)$, caso caso contrário o grau do gene é (1,1).

$$\Theta I : CI \times GI \nrightarrow \mathbb{Q}^+$$

$$[ThetaIPertenceNaoNegativoDef]$$
$$\forall c : CI;\ g : GI \mid g \in c \land (mesmo\mathbb{Q}(grau(g), (0, 1)) \lor mesmo\mathbb{Q}(grau(g), (1, 1))) \bullet \Theta I(c, g) = (0, 1)$$
$$[ThetaIPertenceNegativoDef]$$
$$\forall c : CI;\ g : GI \mid g \in c \land mesmo\mathbb{Q}(grau(g), (1, 1)) \bullet \Theta I(c, g) = (1, 1)$$

A definição do requisito rI para o problema de controle de semáforo usado pela função $selI$ é definido como um subconjunto do conjunto de cromossomos que apresentam pelo menos um semáforo com fluxo lido abaixo do fluxo padrão.

$$rI : \mathbb{P}(PI)$$

$$\forall c : PI \bullet \exists g : GENEI;\ s : SEMAFORO;\ b : BASE \bullet$$
$$g \in c \land b = head\,(g) \land s \in \mathsf{dom}\ b \land FluxoPadra(s) - FludoLido < 0 \land c \in rI$$

A função $fecI$ usada para definir o conjunto de genes dominantes existentes nos cromossomos pais submetidos a operação de cruzamento está correta com o modelo GAADT se ela for reflexiva.

Lema 5.3.7. $\forall c : CI \bullet (fecI(c, c), c) \in\equiv I_C$

5.3.3 Algoritmo

O valor dos parâmetros TI e KI usados na definição da função $GAADT$ como alternativas de parada são definias para o problema do semáforo da seguite maneita.

$$\begin{array}{|l} TI : \mathbb{Z} \\ kI : \mathbb{Q} \\ \hline TI = 100 \wedge \\ kI > (0, 1) \end{array}$$

Neste caso, o algoritmo irá parar quando atingir a iteração 100, ou quando encontrar uma população formada por cromossomos de adaptação maior do que zero. Significanto que o fluxo de veículos em todas as regiões da cidade está funcionando sem problemas

5.4 Monitoramento de Sinais Vitais

Os avanços obtidos em tecnologia, principalmente em computação, mecânica, eletrônica, e obviamente em medicina nos últimos anos vêm contribuindo para o desenvolvimento de novos procedimentos médicos baseados em técnicas de inteligência artificial, de realidade virtual e de banco de dados. A utilização destas técnicas na construção de *software* iterativos têm permitido corrigir e melhorar os procedimentos médicos já existentes, além de aumentar o conhecimento do médico sobre seu paciente, melhorando assim o atendimento clínico.

Técnicas de inteligência artificial são usadas na construção de *software* educativo [71] e de apoio à decisão [87, 128], que armazenam em bases de conhecimentos um conhecimento especializado numa determinada

área. No caso do conhecimento médico sobre uma dada enfermidade, estas técnicas têm como objetivo auxiliar o médico a diagnosticar e tratar da referida enfermidade. Estes *software* recebem como entrada um conjunto de dados sobre a situação do paciente e o diagnóstico do médico para o paciente. Se a decisão tomada pelo médico estiver correta, então o *software* aceita o diagnóstico, caso contrário o *software* inicia um diálogo com o médico questionando sua decisão ou ressaltando a relevância de um dado importante para o diagnóstico em análise. Este diálogo continua até que uma decisão correta para a caso em análise seja obtida, ou até que o médico entre com um novo fato ou dado na base de conhecimento do *software* que altere sua decisão sobre aquela enfermidade; por exemplo a descoberta de uma variação do vírus responsável pela enfermidade em questão, quando for o caso.

Técnicas de realidade virtual são usadas na construção de *software* para auxiliar o treinamento, planejamento e realização de cirurgias. Em [125, 97, 102] são descritos alguns *software* de planejamento cirúrgico. A entrada destes *software* são informações extraídas de exames tais como tomografia computadorizada e ressonância magnética da parte do organismo a ser operada. Com base na entrada fornecida, o *software* cons-trói um organismo virtual para que o médico possa, dentro do ambiente virtual, testar vários procedimentos cirúrgicos e assim planejar a sua estratégia de trabalho. Já em [19, 18], são encontradas descrições de um *software* de telecirurgia que permite que um médico na Alemanha opere um paciente no Brasil. As entradas deste *software* são informações captadas por sensores instalados sobre o organismo do paciente, cujas informações são enviadas para o servidor remoto, convertidas em um organismo virtual, sobre o qual o médico realiza a operação, e suas ações são enviadas de volta a sala cirúrgica, onde braços mecânicos realizam a operação com a ajuda de uma equipe médica.

Nesta seção, uma instanciação do GAADT será usada para monitorar os sinais vitais de pacientes em uma unidade de tratamento semi-intensivo. Porém assim como na seção anterior algums elementos desta especificação serão suprimidos por já terem sido apresentadas nas Tabelas 5.2 e 5.3.

5.4.1 Tipos Básicos

O alfabeto utilizado para a construção do material genético de um sistema de monitoramento de sinais vitais é formado pelo conjunto dos pares (s, n), onde s é um dos sinais monitorados e n é a leitura do respectivo sinal no paciente, e por um conjunto de códigos de doenças ($\{d_1, d_2, d_3, d_4\}$). A este alfabeto é adicionado um código d_λ usado para formar o gene inócuo. Para efeito da especificação desta aplicação será considerado que

os sinais monitorados são: temperatura (t), infusão (i), diuresis (d) e pressão arterial (p).

$$[LEITURA]$$
$$DOENCA ::= \{d_1, d_2, d_3, d_4, d_\lambda\}$$
$$BI ::= LEITURA \cup DOENCA$$

A semântica atribuída ao gene $g = \langle b_1, ..., b_5 \rangle$ é que as leituras $b_1, ..., b_4$ realizadas conduzem ao diagnóstico da doença b_5 resultante da evolução do quadro clínico observado. A construção dos genes deve obedecer ao seguinte conjunto de axiomas:

- para todo gene, os elementos que ocupam as posições referentes às leituras dos sinais monitorados são ocupadas pelos valores pertencentes ao *given set* $LEITURA$;
- para todo gene, o elemento que ocupa a posição relativa ao código da doença é ocupada por valores pertencentes ao conjunto $\{d_1, d_2, d_3, d_4, d_\lambda\}$;
- para todo gene, se o elemento que ocupa a posição relativa ao código da doença for diferente da base inócua, então a probabilidade de ocorrência da referida doença é maior do que 0%.

Antes de ser apresentado o conjunto de axiomas de formação de genes para este exemplo é necessário que se especifique a função que mapeia as configurações dos sinais monitorados a uma probabilidade maior do que 0%.

$prob : LEITURA \times LEITURA \times LEITURA \times LEITURA \times DOENCA \pfun \mathbb{Z}$

$prob((t,50),(i,40),(d,20),(p,70),d_1) = 11 \land prob((t,51),(i,40),(d,20),(p,70),d_1) = 85 \land$
$prob((t,52),(i,40),(d,20),(p,70),d_1) = 74 \land prob((t,53),(i,40),(d,20),(p,70),d_1) = 33 \land$
$prob((t,54),(i,40),(d,20),(p,70),d_1) = 58 \land prob((t,55),(i,40),(d,20),(p,70),d_1) = 66 \land$
$prob((t,56),(i,40),(d,20),(p,70),d_1) = 78 \land prob((t,57),(i,40),(d,20),(p,70),d_1) = 91 \land$
$prob((t,58),(i,40),(d,20),(p,70),d_1) = 20 \land prob((t,59),(i,40),(d,20),(p,70),d_1) = 51 \land$
$prob((t,30),(i,40),(d,20),(p,70),d_2) = 65 \land prob((t,30),(i,38),(d,20),(p,70),d_2) = 57 \land$
$prob((t,30),(i,37),(d,20),(p,70),d_2) = 82 \land prob((t,30),(i,34),(d,20),(p,70),d_2) = 77 \land$
$prob((t,30),(i,33),(d,20),(p,70),d_2) = 40 \land prob((t,30),(i,30),(d,20),(p,70),d_2) = 36 \land$
$prob((t,30),(i,28),(d,20),(p,70),d_2) = 63 \land prob((t,30),(i,27),(d,20),(p,70),d_2) = 18 \land$
$prob((t,30),(i,27),(d,20),(p,70),d_2) = 61 \land prob((t,30),(i,28),(d,20),(p,70),d_2) = 25 \land$
$prob((t,30),(i,40),(d,22),(p,60),d_3) = 92 \land prob((t,30),(i,40),(d,23),(p,59),d_3) = 50 \land$
$prob((t,30),(i,40),(d,22),(p,55),d_3) = 64 \land prob((t,30),(i,40),(d,23),(p,54),d_3) = 80 \land$
$prob((t,30),(i,40),(d,23),(p,53),d_3) = 16 \land prob((t,30),(i,40),(d,23),(p,50),d_3) = 78 \land$
$prob((t,30),(i,40),(d,22),(p,50),d_3) = 40 \land prob((t,30),(i,40),(d,22),(p,49),d_3) = 63 \land$
$prob((t,30),(i,40),(d,22),(p,49),d_3) = 82 \land prob((t,30),(i,40),(d,23),(p,48),d_3) = 77 \land$
$prob((t,31),(i,40),(d,20),(p,60),d_4) = 84 \land prob((t,31),(i,39),(d,20),(p,59),d_4) = 30 \land$
$prob((t,31),(i,48),(d,20),(p,55),d_4) = 61 \land prob((t,32),(i,48),(d,20),(p,54),d_4) = 97 \land$
$prob((t,32),(i,48),(d,20),(p,53),d_4) = 16 \land prob((t,33),(i,47),(d,20),(p,50),d_4) = 57 \land$
$prob((t,33),(i,47),(d,20),(p,50),d_4) = 83 \land prob((t,34),(i,47),(d,20),(p,49),d_4) = 46 \land$
$prob((t,34),(i,46),(d,20),(p,49),d_4) = 85 \land prob((t,34),(i,46),(d,20),(p,48),d_4) = 77$

$AFGI : \mathbb{P}(\ \mathsf{seq}\ BI)$

$(\forall g : AFGI \bullet \#g = 5) \land$
$(\forall g : AFGI \bullet \exists l_1, l_2, l_3, l_4, l_5 : LEITURA \bullet l_1 = g(1) \land l_2 = g(2) \land l_3 = g(3) \land l_4 = g(4)) \land$
$(\forall g : AFGI \bullet \exists d : DOENCA \bullet d = g(5)) \land$
$(\forall g : AFGI \mid g(5) \neq d_\lambda \bullet\ \mathsf{ran}\ g \in\ \mathsf{dom}\ prob)$

O gene inócuo neste caso é qualquer gene cujo código da doença é igual à base inócua.

$$gI_\lambda = \langle (t,0),(i,0),(d,0),(p,0),d_\lambda \rangle$$

O tipo cromossomo para este problema é formado por um único gene. Sendo assim, o conjunto de axiomas de formação de cromossomos para este exemplo é:

$$AFCI : \mathbb{P}(\mathbb{P}(GI))$$
$$\forall c : AFCI \bullet \#c = 1$$

5.4.2 Operadores Genéticos

O grau de adaptação do gene $g = \langle b_1, ..., b_5 \rangle$ é igual à probabilidade de ocorrência da doença b_5 frente a configuração dos sinais monitorados, a qual é definida como:

$$grauI : GI \nrightarrow \mathbb{Q}$$
$$[GrauIInocuoDef]$$
$$grauI(gI_\lambda) = (0,1)$$
$$[GrauINaoInocuoDef]$$
$$\forall g : GI \mid g \neq gI_\lambda \bullet grauI(g) = (prob(g),1)$$

Lema 5.4.1. $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), grauI(gI_\lambda)) \in maior\mathbb{Q}$.

O atributo relevante para o monitoramento dos sinais vitais é o conjunto das leituras consideradas, pois o objetivo deste sistema é encontrar a doença que apresente a maior probabilidade para um dado conjunto de leituras considerado.

$$
\begin{array}{|l}
atributoRelevanteI : \mathbb{P}(\mathbb{P}(GI)) \\
\hline
[atributoRelevanteIDef] \\
\forall g_1, g_2 : GI \bullet (g_1(1) = g_2(1) \land g_1(2) = g_2(2) g_1(3) = g_2(3) \land g_1(4) = g_2(4)) \Leftrightarrow \\
(\exists G_1 : atributoRelevanteI \bullet \{g_1, g_2\}) \subseteq G_1)
\end{array}
$$

A relação $mesmaI$ considerando a especificação acima do conjunto $atributoRelevanteI$ deve satisfazer as propriedades reflexiva, comutativa e transitiva como exigido na definição da relação $mesma$ do GAADT.

Lema 5.4.2. $\forall g : GI \bullet (g, g) \in mesmaI$

Lema 5.4.3. $\forall g_1, g_2 : GI \mid (g_1, g_2) \in mesmaI \bullet (g_2, g_1) \in mesmaI$

Lema 5.4.4. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$

Para verificar se a especificação apresentada até o momento está de acordo com a definição da função gene dominante, deve-se exigir a prova da propriedade reflexiva e associativa da função $domi$.

Lema 5.4.5. $\forall g : GI \mid g = gI_\lambda \bullet domiI(g, g) = g \Leftrightarrow (g, g) \in mesmaI$

Lema 5.4.6. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_1, g_3) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet$
$domiI(g_1, domiI(g_2, g_3)) = domiI(domiI(g_1, g_2), g_3)$

O peso de um gene g no cromossomo c para o problema de monitoramento de sinais vitais é igual a $(1, 1)$.

$$
\begin{array}{|l}
\Theta I : CI \times GI \nrightarrow \mathbb{Q}^+ \\
\hline
[ThetaINaoPertenceDef] \\
\forall c : CI;\ g : GI \mid g \notin c \bullet \Theta I(c, g) = (0, 1) \\
[ThetaIPertenceDef] \\
\forall c : CI;\ g : GI \mid g \in c \bullet \Theta I(c, g) = (1, 1)
\end{array}
$$

A comprovação de que a especificação até este ponto da instanciação do GAADT está correta pode ser obtida pela prova da propriedade reflexiva da função $fecI$.

Lema 5.4.7. $\forall c : CI \bullet (fecI(c, c), c) \in\equiv I_C$

Os predicados MI e FI serão considerados vazios porque nesta instanciação a execução da operação de cruzamento não contribuirá para a evolução da população trabalhada pelo GAADT.

$$\begin{array}{|l} \\ MI : \mathbb{P}(PI) \\ FI : \mathbb{P}(PI) \\ \hline MI = \emptyset \wedge \\ FI = \emptyset \\ \end{array}$$

5.4.3 Algoritmo

Os valores adotados para as duas primeiras alternativas de parada na definição da função *GAADTI*, e referidos como número máximo de iterações e adaptação do cromossomo considerada satisfatória para o problema em foco, são definidos abaixo.

$$\begin{array}{|l} \\ TI : \mathbb{Z} \\ kI : \mathbb{Q} \\ \hline TI = 100 \wedge \\ kI = (75, 1) \\ \end{array}$$

Neste caso, como no exemplo do problema do caixeiro viajante, o algoritmo poderá parar quando atingir a iteração 100 ou quando encontrar um cromossomo com adaptação maior ou igual a 75.

A população inicial considerada é formada por um único cromossomo formado pela leitura atual dos sinais monitorados conduzindo a nenhuma doença $\{\{\langle (t, 30), (i, 28), (d, 20), (p, 70), d_\lambda \rangle\}\}$, o qual irá evoluir através da ação da operação de mutação para uma doença com probabilidade de ocorrer frente às leituras atuais acima de 75, se existir tal doença, caso contrário o sistema continuará monitorando o paciente até que uma nova leitura diferente da atual seja fornecida. Neste momento a nova leitura será atribuída à variável população inicial, e o processo de evolução se repete.

Com base na instanciação do GAADT para o sistema de monitoramento de sinais vitais será mostrado na próxima seção, que $GAADTI$ é um refinamento de $GAADT$.

5.5 Refinamento

O refinamento apresentado aqui pode ser aplicado para todas as instanciações do GAADT apresentadas, independente da natureza do problema a ser resolvido. Para tanto, considere que os tipos básicos do $GAADT$ são instanciados pelos tipos básicos definidos para o problema em questão, no caso o problema de construção de árvores filogenéticas ou o sistema de monitoramento de sinais vitais, e que as definições axiomáticas de $GAADTI$ satisfazem as propriedades originais. Pela Definição 5.1.2, diz-se que $GAADTI$ é um refinamento de $GAADT$ se os teoremas de inicialização, aplicabilidade e corretude forem satifeitos. Porém, antes de apresentar a prova destes teoremas é necessário considerar a instanciação como um mapeamento entre os tipos e apresentar o esquema $Retrieve$ que é a função identidade.

Considere o seguinte mapeamento (interpretação) dos tipos abstratos nas instanciações $Instantiate = \{B \to BI, AFG \to AFGI, AFC \to AFCI, atributoRelevante \to atributoRelevanteI, M \to MI, F \to FI, grau \to grauI, \Theta \to \Theta I, g_\lambda \to gI_\lambda, p_{corte} \to pI_{corte}, P_0 \to PI_0, k \to kI, T \to TI\}$. Como o estado da instanciação preserva a estrutura do estado do GAADT, o $Retrieve$ transforma-se na função identidade.

$Retrieve$

$Instantiate$
$StateGAADT$
$StateGAADTI$

$Tx = TxI$
$\wedge P_{atual} = PI_{atual} \wedge P_{cruz} = PI_{cruz} \wedge P_{mut} = PI_{mut}$
$\wedge t = tI$
$\wedge MACHO = MACHOI \wedge FEMEA = FEMEAI$

No esquema acima, $Instantiate$ é usado como parâmetros da função $Retrieve$, pois os componentes do estado não podem ser comparados diretamente.

Lema 5.5.1. $InitGAADTI \wedge Retrieve' \implies InitGAADT$

Lembre-se que o esquema $GAADT$ é definido como a disjunção do esquema $Termina$ com a composição entre a conjunção dos esquemas $Cruzamento$ e $Mutacao$ com o esquema $Iteracao$. O esquema $Termina$, por sua vez, é definido como a disjunção dos esquemas $TerminaPorAdaptacao$ e $TerminaPorTempo$. Mas como estas operações de esquemas não são monotônicas em relação ao refinamento [72], então para provar que a operação $GAADTI$ é um refinamento da operação $GAADT$ têm-se duas opções. A primeira é construir um esquema simples para cada uma destas operações através da eliminação dos operadores de esquemas e depois provar que há um refinamento entre estas operações [126]. A segunda opção consiste em aplicar as leis propostas por Groves [72], que descrevem algumas situações particulares nas quais os operadores de conjunção e disjunção de esquemas podem ser considerados monotônicos para o refinamento, definidas em função dos conceitos de aplicabilidade, corretude e consistência entre dois esquemas de operação. O refinamento apresentado nesta tese do GAADT está de acordo com a segunda opção.

Dados dois esquemas, diz-se que eles são consistentes se a sua conjunção é definida sobre o estado de dados em que ambos os esquemas forem definidos.

Definição 5.5.2. (Consistência) - Seja W e Y dois esquemas de operação que agem sobre os estados $StateW$ e $StateY$, com esquema de inicialização $InitW$ e $InitY$, para os quais o Teorema de Inicialização é válido (ver Definição 5.1.2). Diz-se que o esquema W é consistente com o esquema Y, se e somente se, $\forall Retrieve \bullet$ $\mathsf{pre}\ W \wedge \mathsf{pre}\ Y \implies \mathsf{pre}\ (W \wedge Y)$.

Dado um esquema X, definido como a conjunção dos esquemas W e Y, e sua versão concreta $X_1 = W_1 \wedge Y_1$, onde W_1 e Y_1 são as versões concretas dos esquemas W e Y, respectivamente, diz-se que X_1 é um refinamento de X, se o esquema W_1 for um refinamento de W, Y_1 for um refinamento de Y, e a conjunção do esquema W e Y for consistente com a conjunção das suas versões concretas.

Definição 5.5.3. (Refinamento de Conjunção de Esquemas) - Sejam W e Y dos esquemas de operação, e W_1 e Y_1 versões concretas destes esquemas. Diz-se que $W_1 \wedge Y_1$ é um refinamento de $W \wedge Y$, se as seguintes propriedades forem verdadeiras:

1. W_1 é um refinamento de W,
2. Y_1 é um refinamento de Y,
3. $\forall Retrieve \bullet \mathsf{pre}\ (W \wedge Y) \wedge \mathsf{pre}\ (W_1 \wedge Y_1) \implies \mathsf{pre}\ ((W \wedge W_{1)} \wedge (Y \wedge Y_1))$.

Dado um esquema X, definido como a disjunção dos esquemas W e Y, e sua versão concreta $X_1 = W_1 \vee Y_1$, onde W_1 e Y_1 são as versões concretas dos esquemas W e Y, respectivamente, diz-se que X_1 é um refinamento de X, se o esquema W_1 for um refinamento de W, Y_1 for um refinamento de Y, $W \vee Y$ for consistente com Y_1 nas entradas da pré-condição de W, e $W \vee Y$ for consistente com W_1 nas entradas da pré-condição de Y.

Definição 5.5.4. (Refinamento de Disjunção de Esquemas) - Sejam W e Y dos esquemas de operação, e W_1 e Y_1 versões concretas destes esquemas. Diz-se que $W_1 \vee Y_1$ é um refinamento de $W \vee Y$, se as seguintes propriedades forem verdadeiras:

1. W_1 é um refinamento de W,
2. Y_1 é um refinamento de Y,
3. $\forall Retrieve \bullet \text{ pre } W \wedge Y_1 \implies W \vee Y$, e
4. $\forall Retrieve \bullet \text{ pre } Y \wedge W_1 \implies W \vee Y$.

O refinamento da composição de dois esquemas pode ser obtido pela eliminação do operador de composição combinada com a Definição 5.5.3. A eliminação do operador de composição de dois esquemas de operação W e X que agem sobre o mesmo estado de dados pressupõe a existência de um estado de dados $State$", que é uma configuração do estado de dados após a ação da operação W, a qual corresponde uma configuração do estado de dados antes da operação X ser requerida. Inicialmente, procede-se a substituição do estado $State'$ na operação W pelo estado $State$" e do estado $State$ na operação X pelo estado $State$", para depois fazer-se o combinação da parte declarativa e a conjunção da parte predicativa dos esquemas obtidos por estas substituições [126].

Definição 5.5.5. (Eliminação da Composição) - Sejam $W = [\Delta State;\ i_W? : I;\ o_W! : O \mid PredicadoW]$ e $X = [\Delta State;\ i_X? : I;\ o_X! : O \mid PredicadoX]$ dois esquemas de operação, então o esquema $W \ {}^{0}_{9}\ X = [\Delta State;\ i_W?, i_X? : I;\ o_W!, o_X! : O \mid \exists State" \bullet PredicadoW[State'/State"] \wedge PredicadoX\ [State/State"]]$.

De posse desta definição diz-se que o esquema $W_1 \ {}^{0}_{9}\ Y_1$ é um refinamento de $W \ {}^{0}_{9}\ Y$, onde W_1 e Y_1 são versões concretas dos esquemas W e Y respectivamente, se existirem os estados $State$" e $State_1$" tal que o esquema $W_1[State'/State"] \ {}^{0}_{9}\ Y_1[State/State"]$ refina o esquema $W[State'_1/State_1"] \ {}^{0}_{9}\ Y[State_1/State_1"]$.

Definição 5.5.6. (Refinamento de Composição de Esquemas) - Sejam W e Y dos esquemas de operação, e W_1 e Y_1 versões concretas destes esquemas. Diz-se que $W_1 \ {}^{0}_{9} \ Y_1$ é um refinamento de $W \ {}^{0}_{9} \ Y$, se as seguintes propriedades forem verdadeiras:

1. W_1 é um refinamento de W,
2. Y_1 é um refinamento de Y,
3. $\exists State"; \ State_1" \bullet \forall Retrieve \bullet$
   $\mathsf{pre} \ (W[State'/State"] \wedge Y[State/State"]) \wedge \mathsf{pre} \ (W_1[State'_1/State_1"] \wedge Y_1[State_1/State_1"]) \implies$
   $\mathsf{pre} \ ((W[State'/State"] \wedge W_1[State'_1/State_1"]) \wedge (Y[State/State"] \wedge Y_1[State_1/State_1"])).$

Lema 5.5.7. *$TerminaPorAdaptacaoI$ é um refinamento de $TerminaPorAdaptacao$, ou seja, os seguintes teoremas são válidos:*

- $\mathsf{pre} \ TerminaPorAdaptacao \wedge Retrieve \implies \mathsf{pre} \ TerminaPorAdaptacaoI$
- $\mathsf{pre} \ TerminaPorAdaptacao \wedge Retrieve \wedge TerminaPorAdaptacaoI \wedge Retrieve' \implies TerminaPorAdaptacao$

Lema 5.5.8. *$TerminaPorTempoI$ é um refinamento de$TerminaPorTempo$, ou seja, os seguintes teoremas são válidos:*

- $\mathsf{pre} \ TerminaPorTempo \wedge Retrieve \implies \mathsf{pre} \ TerminaPorTempoI$
- $\mathsf{pre} \ TerminaTempo \wedge Retrieve \wedge TerminaTempoI \wedge Retrieve' \implies TerminaTempo$

Lema 5.5.9. *$TerminaI$ é um refinamento de $Termina$, ou seja, os seguintes teoremas são válidos:*

- *$TerminaPorAdaptacaoI$ é um refinamento de $TerminaPorAdaptacao$*
- *$TerminaPorTempoI$ é um refinamento de $TerminaPorTempo$*
- $\forall Retrieve \wedge Retrieve' \wedge \mathsf{pre} \ TerminaPorAdaptacao \wedge TerminaPorTempoI \implies TerminaPorAdaptacao \vee TerminaPorTempo,$

- $Retrieve \wedge Retrieve' \wedge$ pre $TerminaPorTempo \wedge TerminaPorAdaptacaoI \implies TerminaPorTempo \vee TerminaPorAdaptacao$

Lema 5.5.10. *$CruzamentoI$ é um refinamento de $Cruzamento$, ou seja, os seguintes teoremas são válidos:*

- pre $Cruzamento \wedge Retrieve \implies CruzamentoI$
- pre $Cruzamento \wedge Retrieve \wedge CruzamentoI \wedge Retrieve' \implies Cruzamento$

Lema 5.5.11. *$MutacaoI$ é um refinamento de $Mutacao$, ou seja, os seguintes teoremas são válidos:*

- pre $Mutacao \wedge Retrieve \implies$ pre $MutacaoI$
- pre $Mutacao \wedge Retrieve \wedge MutacaoI \wedge Retrieve' \implies Mutacao$

Lema 5.5.12. *$CruzamentoI \wedge MutacaoI$ é um refinamento de $Cruzamento \wedge Mutacao$, ou seja, os seguintes teoremas são válidos:*

- *$CruzamentoI$ é um refinamento de $Cruzamento$*
- *$MutacaoI$ é um refinamento de $Mutacao$*
- $\forall Retrieve \wedge Retrieve' \wedge$ pre $(Cruzamento \wedge Mutacao) \wedge$ pre $(CruzamentoI \wedge MutacaoI) \implies$ pre $((Cruzamento \wedge CruzamentoI) \wedge (Mutacao \wedge MutacaoI))$

Lema 5.5.13. *$IteracaoI$ é um refinamento de $Iteracao$, ou seja, os seguintes teoremas são válidos:*

- pre $Iteracao \wedge Retrieve \implies$ pre $IteracaoI$
- pre $Iteracao \wedge Retrieve \wedge IteracaoI \wedge Retrieve' \implies Iteracao$

Lema 5.5.14. *$(CruzamentoI \wedge MutacaoI)\ {}^{0}_{9}\ IteracaoI$ é um refinamento de $(Cruzamento \wedge Mutacao)\ {}^{0}_{9}\ Iteracao$, ou seja, os seguintes teoremas são válidos:*

- *$CruzamentoI \wedge MutacaoI$ é um refinamento de $Cruzamento \wedge Mutacao$*
- *$IteracaoI$ é um refinamento de $Iteracao$*

- $\exists StateGAADT''; \ StateGAADTI'' \bullet \forall Retrieve \wedge Retrieve' \wedge$ pre $((Cruzamento \wedge Mutacao)$ $[StatGAADT'/StateGAADT''] \wedge Iteracao[StatGAADT/StateGAADT''])\wedge$ pre $((CruzamentoI \wedge$ $MutacaoI)[StatGAADTI'/StateI''] \wedge IteracaoI[StatGAADTI/StateGAADTI'']) \implies$ pre $((Cruzamento \wedge Mutacao)[StatGAADT'/StateGAADT''] \wedge (CruzamentoI \wedge MutacaoI)$ $[StatGAADTI'/StateGAADTI'']) \wedge (Iteracao[StatGAADT/StateGAADT''] \wedge IteracaoI$ $[StatGAADTI/StateGAADTI'']))$

Lema 5.5.15. *$GAADTI$ é um refinamento de $GAADT$, ou seja, os seguintes teoremas são válidos:*

- $(CruzamentoI \wedge MutacaoI) \ {}^{0}_{9} \ IteracaoI$ *é um refinamento de* $(Cruzamento \wedge Mutacao) \ {}^{0}_{9} \ Iteracao$
- $TerminaI$ *é um refinamento de* $Termina$
- $\forall Retrieve \wedge Retrieve' \wedge$ pre $Termina \wedge (CruzamentoI \vee MutacaoI) \ {}^{0}_{9} \ IteracaoI \implies Termina \vee$ $(Cruzamento \vee Mutacao) \ {}^{0}_{9} \ Iteracao$
- $\forall Retrieve \wedge Retrieve' \wedge Termina \wedge$ pre $((CruzamentoI \vee MutacaoI) \ {}^{0}_{9} \ IteracaoI) \implies Termina \vee$ $(Cruzamento \vee Mutacao) \ {}^{0}_{9} \ Iteracao$

Capítulo 6

Resultados

6.1 Introdução

O dilema da qualidade do resultado versus o número de iterações necessárias para um algoritmo genético convergir é uma das barreiras encontradas por muitos pesquisadores ao tentar adotar este algoritmo como uma abordagem computacional para solucionar problemas [110], já que o número de iterações executadas por um algoritmo genético está diretamente relacionado à qualidade do resultado encontrado. Se o critério de parada via número de iterações de um algoritmo genético for inferior ao número de iterações necessário para este algoritmo ser bem sucedido, então o cromossomo retornado não irá satisfazer às exigências mínimas de qualidade requeridas para o problema. Outros fatores citados na literatura como responsáveis pelo mau desempenho dos algoritmos genéticos são: a população inicial, o tamanho da população, o comportamento das operações de cruzamento e mutação, e os critérios adotados pelas operações de seleção e substituição.

A população inicial influencia a convergência do algoritmo por ser ela a fonte dos cromossomos que devem ser melhorados para atender às exigências de qualidade do problema. Quando os cromossomos desta população não apresentarem alguma das características presentes nos resultados procurados para o problema, o algoritmo genético precisará de mais iterações para convergir do que no caso de existir pelo menos um cromossomo na população inicial que apresente parte das características procuradas.

A representação da população por uma estrutura de dados de tamanho fixo é responsável pela perda das características consideradas relevantes para o problema em análise, a qual ocorre sempre que o número de cromossomos adaptados ao ambiente na população atual, adicionado ao número de cromossomos adaptados na população de cromossomos gerados pela ação dos operadores de cruzamento e mutação, é maior do que o tamanho permitido para a população. Logo, se as características exigidas para o resultado do problema

forem perdidas pela restrição de tamanho imposta pela representação da população adotada, então o algoritmo genético irá precisar de um maior número de iterações para convergir.

A operação de cruzamento de um algoritmo genético é responsável por combinar as características dos indivíduos pais com o objetivo de gerar novos indivíduos, que podem ser mais ou menos adaptados do que os indivíduos pais, dependendo das características combinadas. O comportamento desta operação está diretamente relacionado com a representação dos indivíduos e do tipo de problema em análise.

O comportamento da operação de seleção e substituição pode acelerar ou retardar o processo de convergência de um algoritmo genético, dependendo do problema, e em decorrência requerer um número maior ou menor de iterações para o algoritmo genético convergir. Uma evolução retardada é adequada sempre que o número de características não desejadas para o resultado do problema for predominante entre duas populações sucessivas.

Neste capítulo, será apresentado um estudo comparativo baseado no valor da adaptação média e da melhor adaptação encontrada em cada iteração, entre o GAADT, o algoritmo genético baseado em aprendizagem por *linkage* e o algoritmo genético de tempo contínuo, quando aplicado ao problema do caixeiro viajante [42] e da construção de árvores filogenéticas. E será apresentado também um estudo comparativo entre o tempo requerido por uma implementação em *software* do GAADT versus uma em *hardware*, já que o tempo de convergência neste caso é um fator determinante para a escolha da abordagem a ser adotada, muito embora não façam parte da motivação apresentada para o desenvolvimento do GAADT questões de eficiência. A ausência de um estudo comparativo entre a implementação do GAADT para o problema de monitoramento com um outro modelo de algoritmo genético deve-se ao fato de que este problema exige um processamento em tempo real, no qual as alterações no quadro clínico do paciente possam se refletir quase que instantaneamente na evolução da população manipulada pelo algoritmo. No GAADT o tratamento de problemas de tempo real só é possível porque este recebe como entrada o ambiente e não só a população, de modo que qualquer alteração no ambiente é imediatamente considerada na sua execução. No caso dos outros modelos de algoritmo genético seria preciso interromper a execução atual do algoritmo para que tais alterações fossem consideradas, já que as leituras dos sinais monitorados nestes algoritmos são o alfabeto sobre o qual o cromossomo é construído.

6.2 Caixeiro Viajante

A presente seção relata a média de 100 resultados empíricos obtidos por uma implementação na linguagem C do GAADT aplicado ao problema do caixeiro viajante, descrito no Capítulo 3, do algoritmo genético baseado em aprendizagem por *linkage* (Seção 2.3.3) e do algoritmo genético de tempo contínuo (Seção 2.4.2). Para esta análise foi considerado que o número de cidades a serem visitadas pelo caixeiro é 50. O algoritmo genético de aprendizagem baseado em *linkage* utiliza probabilidade de cruzamento de 65%; seleção elitista com $h = 20$ e substituição elitista com $l = 20$. Enquanto o algoritmo genético de tempo contínuo adota probabilidade de cruzamento de 65%; probabilidade de mutação de 7%; população representada por um vetor de cromossomos de dimensão igual à 100; cromossomos representados por vetores alfanuméricos sem repetição; ordenado; mutação de troca entre todas as posições; seleção elitista com $h = 20$; substituição elitista com $l = 20$; e operação de continuidade aplicada sobre os 50 cromossomos menos adaptados. Em todos os algoritmos investigados o critério de parada adotado foi o número de iterações igual à 100 e a mesma população inicial foi considerada. Os programas das implementações consideradas nestas Seção podem ser encontrados no seguinte endereço www.mal.ceca.ufal.br.

As figuras 6.1, 6.2 e 6.3 apresentam a evolução da adaptação média da população e do cromossomo menos adaptado para o GAADT, algoritmo genético baseado em aprendizagem por *linkage* e algoritmo genético de tempo contínuo respectivamente.

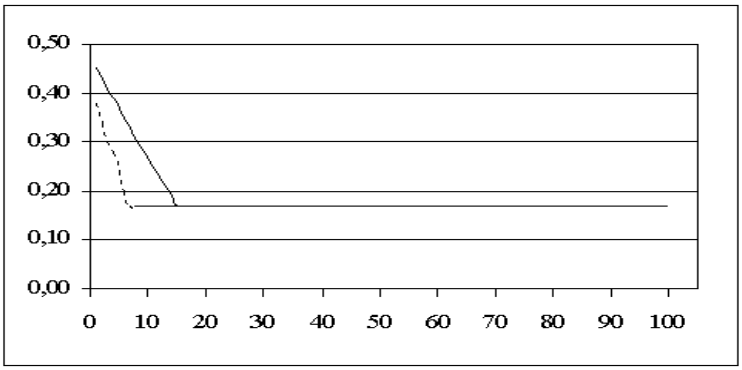


Figura 6.1: Convergência do GAADT para o caixeiro viajante

Como pode ser observado pelo gráfico desenhado nas Figuras 6.1, 6.2 e 6.3 constata-se que o GAADT encontra os cromossomos mais adaptados praticamente na 10 iteração, ficando sua população constituída por somente estes cromossomos a partir da 20 iteração; o algoritmo genético baseado em aprendizagem por *linkage*

encontra os cromossomos mais adaptado na 23 iteração, ficando sua população restrita a estes cromossomos a partir da 25 iteração; e o algoritmo genético de tempo contínuo encontra os cromossomos mais adaptados na 23 iteração; ficando sua população em cada iteração mais ou menos diversificada em função da perturbação gerado pelo operador de continuidade.

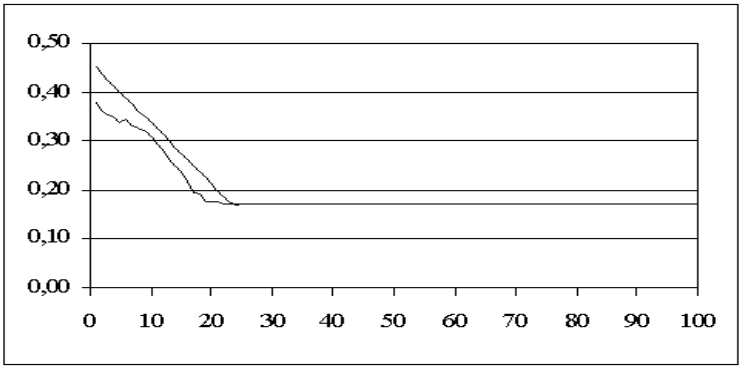


Figura 6.2: Convergência do algoritmo genético baseado em aprendizagem por *linkage* para o caixeiro viajante

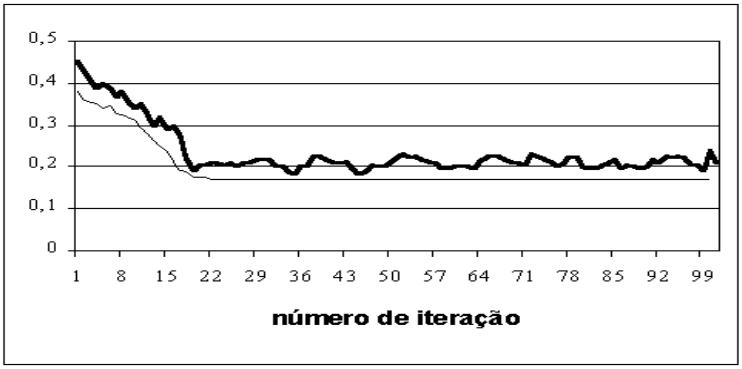


Figura 6.3: Convergência do algoritmo genético de tempo contínuo para o caixeiro viajante

6.3 Controle de Semáforo

Os mesmos algoritmos descritos na seção anterior foram aplicados para o sistema de semáforo da cidade imaginária ilustrada na Figura 6.4. Nesta figura os quarteirões estão representados prelos retângulos escuros, os semáforos pelas circunferências, o sentido do trânsito pelas setas e o semáforo com congestionamento está marcado com um X. Considerou-se para esta implementação que todos os semáforos da rua horizontal que atravessa a cidade apresentam os mesmos tempos para as cores verde, amarelo e verde e os mesmos limites

máximos e mínimos para estas cores, enquanto todos os semáforos localizados nas ruas que atravessam a cidade na vertical compartilham dos mesmos tempos e limites para permanecer nas cores verde, amarela e vermelha. As figuras abaixo apresentam as curvas de convergências do GAADT, do algoritmo genético baseado em aprendizagem por linkage e do algoritmo genético de tempo contínuo.

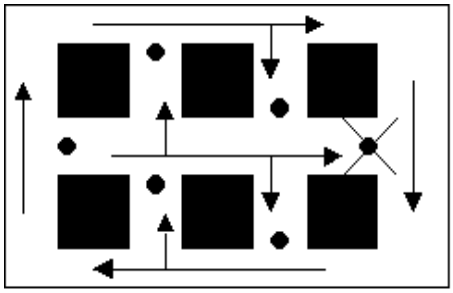


Figura 6.4: Mapa da cidade

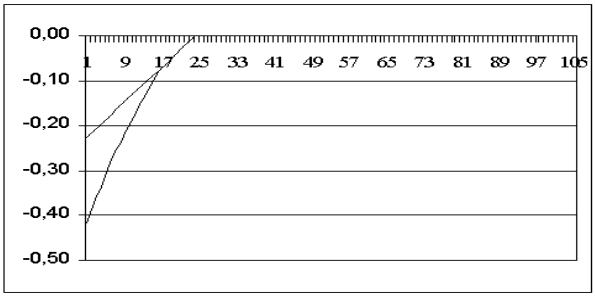


Figura 6.5: Convergência do GAADT para o controle de semáforo

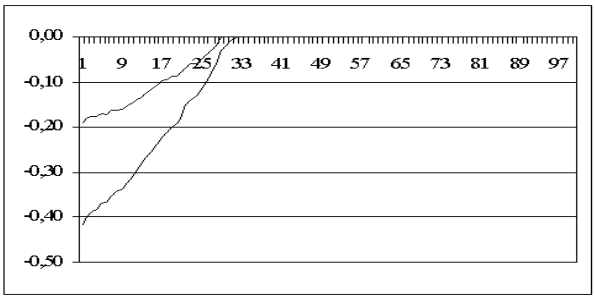


Figura 6.6: Convergência do algoritmo genético baseado em aprendizagem por *linkage* para o controle de semáforo

Como pode ser observado pelos gráficos das Figuras 6.5, 6.6 e 6.7 os resultados obtidos foram semelhantes aos do problema do caixeiro viajante.

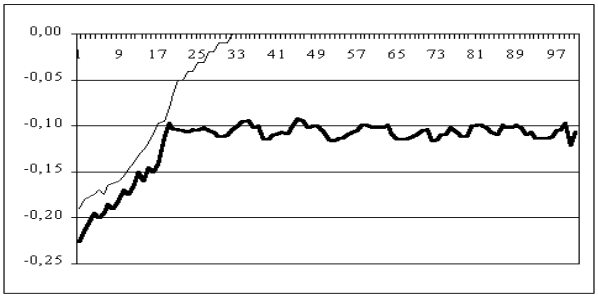


Figura 6.7: Convergência do algoritmo genético de tempo contínuo para o controle do semáforo

6.4 Árvore filogenética

Esta seção apresenta um estudo comparativo dos resultados gerados por uma implementação na linguagem C do GAADT (como especificado no Capítulo 5) do algoritmo genético baseado em aprendizagem por *linkage* e do algoritmo genético de tempo contínuo aplicado ao problema de construção de árvores filogenéticas. Os valores dos parâmetros utilizados pelos algoritmos acima citados são: a probabilidade de cruzamento de 65%; a probabilidade de mutação de 7%; a probabilidade de continuidade 100%; e o número de iterações igual à 100. As operações genéticas adotadas nestas implementações foram: seleção elitista com $h = 20$; substituição elitista com $l = 20$; cruzamento por derivação; e mutação de continuidade aplicada aos 50 cromossomos menos adaptados da população atual. A representação adotada para a população é um vetor de cromossomos; os quais por sua vez são representados por um vetor derivado de uma gramática. A população inicial é formada somente pela árvore filogenética da Figura 5.2. Os programas das implementações consideradas nestas Seção podem ser encontrados no seguinte endereço www.ufpa.br/ccen/roberta/doutorado/programas.

A gramática de construção dos cromossomos para o problema de construção de árvores filogenéticas é apresentada a seguir, a regra de produção inicial desta gramática é a R.

$$
\begin{aligned}
R = \quad & (e_2, e_1, E_{e_3,e_4,e_5}) \mid \quad (e_3, e_1, E_{e_2,e_4,e_5}) \mid \quad (e_4, e_1, E_{e_2,e_3,e_5}) \mid \quad (e_5, e_1, E_{e_2,e_3,e_4}) \\
& (e_1, e_2, E_{e_3,e_4,e_5}) \mid \quad (e_3, e_2, E_{e_1,e_4,e_5}) \mid \quad (e_4, e_2, E_{e_1,e_3,e_5}) \mid \quad (e_5, e_2, E_{e_1,e_3,e_4}) \\
& (e_2, e_3, E_{e_1,e_4,e_5}) \mid \quad (e_1, e_3, E_{e_2,e_4,e_5}) \mid \quad (e_4, e_3, E_{e_2,e_1,e_5}) \mid \quad (e_5, e_3, E_{e_2,e_1,e_4}) \\
& (e_2, e_4, E_{e_3,e_1,e_5}) \mid \quad (e_3, e_4, E_{e_2,e_1,e_5}) \mid \quad (e_1, e_4, E_{e_2,e_3,e_5}) \mid \quad (e_5, e_4, E_{e_2,e_3,e_1}) \\
& (e_2, e_5, E_{e_3,e_1,e_4}) \mid \quad (e_3, e_5, E_{e_2,e_1,e_4}) \mid \quad (e_1, e_5, E_{e_2,e_3,e_4}) \mid \quad (e_4, e_5, E_{e_2,e_3,e_1})
\end{aligned}
$$

$$
\begin{aligned}
E_{e_3,e_4,e_5} &= (e_3,e_4,e_5) \mid (e_4,e_3,e_5) \mid (e_3,e_5,e_4)\\
E_{e_2,e_4,e_5} &= (e_2,e_4,e_5) \mid (e_4,e_2,e_5) \mid (e_2,e_5,e_4)\\
E_{e_2,e_3,e_5} &= (e_2,e_3,e_5) \mid (e_3,e_2,e_5) \mid (e_2,e_5,e_3)\\
E_{e_2,e_3,e_4} &= (e_2,e_3,e_4) \mid (e_3,e_2,e_4) \mid (e_2,e_4,e_3)\\
E_{e_1,e_4,e_5} &= (e_1,e_4,e_5) \mid (e_4,e_1,e_5) \mid (e_1,e_5,e_4)\\
E_{e_1,e_3,e_5} &= (e_1,e_3,e_5) \mid (e_3,e_1,e_5) \mid (e_1,e_5,e_3)\\
E_{e_1,e_3,e_4} &= (e_1,e_3,e_4) \mid (e_3,e_1,e_4) \mid (e_1,e_4,e_3)\\
E_{e_3,e_4,e_5} &= (e_3,e_4,e_5) \mid (e_4,e_3,e_5) \mid (e_3,e_5,e_4)
\end{aligned}
$$

Deve-se observar que a gramática adotada evita a construção de árvores homólogas, esta escolha foi feita para evitar a possível superioridade do GAADT que representa a árvore filogenética como a justaposição de árvores elementares, a qual não faz distinção entre as representações homólogas de uma dada árvore filogenética.

O gráfico que relaciona a adaptação média e o valor da maior adaptação encontrada em cada iteração é ilustrado na figura 6.8 para o GAADT, 6.9 para o algoritmo genético baseado em aprendizagem por *linkage* e 6.10 para o algoritmo genético de tempo contínuo.

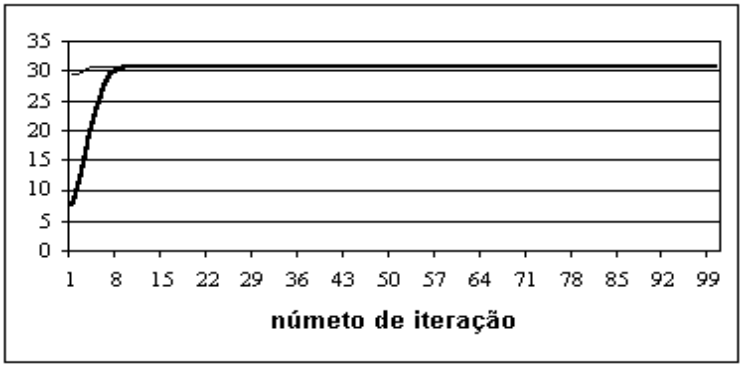


Figura 6.8: Convergência do GAADT para a árvore filogenética

Analisando o gráfico das figuras 6.8, 6.9 e 6.10 verifica-se que o GAADT encontra o cromossomo mais adaptado praticamente na mesma iteração que o algoritmo genético baseado em aprendizagem por *linkage* e o algoritmo genético contínuo. Mas o valor deste cromossomo no caso do algoritmo genético baseado em aprendizagem por *linkage* é menor do que o valor encontrado pelos outros algoritmos, isto acontece porque a população fornecida é composta por uma única árvore filogenética a qual não é alterada pela função de cruzamento deste algoritmo.

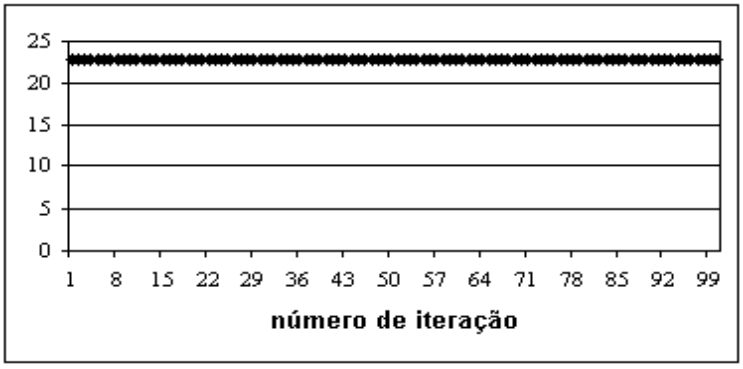


Figura 6.9: Convergência do algoritmo genético baseado em aprendizagem por *linkagem* para a árvore filogenética

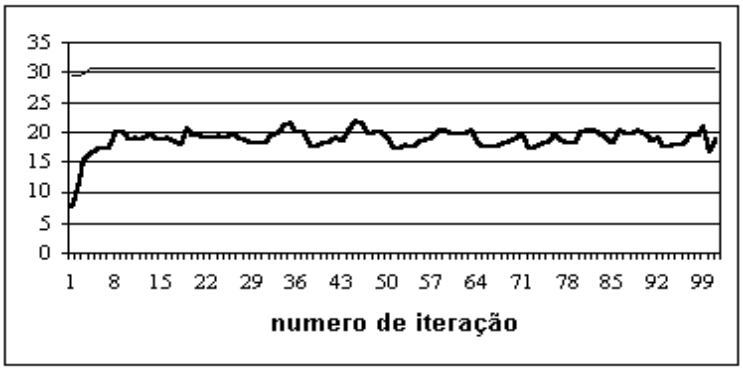


Figura 6.10: Convergência do algoritmo genético de tempo contínuo para a árvore filogenética

6.5 Monitoramento de Sinais de Pacientes na UTI

Uma implementação em *hardware* do algoritmo genético baseado em tipos abstratos de dados será apresentada para monitorar os sinais vitais de um paciente em uma UTI. A escolha desta forma de implementação é respaldada na necessidade de reduzir o tempo de convergência deste algoritmo a um valor aceitável para a velocidade requerida pelo médico na análise dos sinais monitorados. A combinação destas duas abordagens pode ser adequada à solução de problemas de tempo real que necessitem de uma interação constante entre o ambiente e o sistema.

O sistema em questão foi implementado na plataforma de prototipação Chameleon [108], a qual é composta por um componente de *hardware* (FPGA - *Field Programmable Gate Array*) e um componente de *software* (microcontrolador compatível com a família 8051). Estes componentes compartilham bancos de memória, que fornecem espaço de armazenamento de configuração do FPGA, sistema monitor, programa do usuário e canais

de comunicação, como mostra a Figura 6.11.

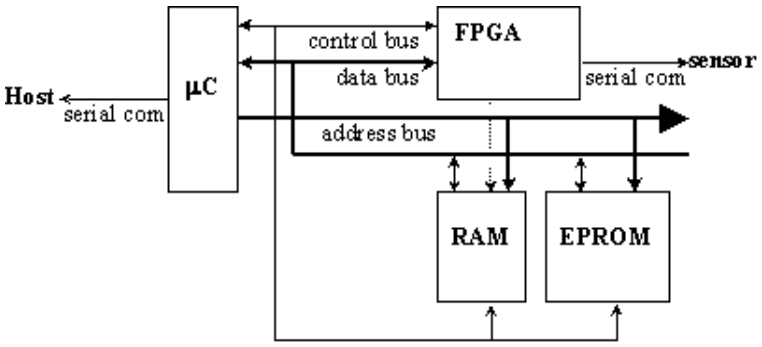


Figura 6.11: Plataforma Chameleon

O microcontrolador será usado para monitorar o FPGA através de um sistema de guarda (*watchdog* entre o *host* e o microcontrolador) e para controlar as comunicações via barramentos de dados e endereços entre memórias e FPGAs. O microcontrolador é também usado para reconfigurar o FPGA em caso de perda de configuração e para buscar informações na base de dados dos pacientes. Todos os programas residentes no microcontrolador foram desenvolvidos na linguagem C e compilados no *software* da Keil.

A base de dados dos pacientes é residente no *host*. Este *host* pode ser conectado a uma rede intranet hospitalar, permitindo assim que o médico possa acompanhar a evolução do quadro clínico de seu paciente a partir de qualquer ponto do hospital ligado à rede.

No FPGA é mapeado, além do algoritmo genético, toda a interface necessária à comunicação com os equipamentos médicos. Estes equipamentos são sensores que medem os sinais monitorados. O FPGA pode ser reconfigurável por programação a diferentes aplicações. Isto o torna um dispositivo ideal para a prototipação rápida de sistemas digitais e uma solução para sistemas dinâmicos [139].

O algoritmo genético especificado na Seção 5.3 foi implementado em um FPGA da família XC4010E da Xilinx. Em tal implementação, a população é formada por um único cromossomo, que por sua vez é constituído por um único gene representado por uma arquitetura codificada da seguinte maneira: 7 bits para cada uma das leituras realizadas e 7 bits para o código da doença possível de ocorrer de acordo com o quadro clínico observado. O diagrama de blocos do FPGA implementado para esta aplicação específica do GAADT é mostrado na Figura 6.12.

O bloco **MUX2x1** recebe o endereço do **Ambiente** constituído pela leitura proveniente dos sensores instalados nos aparelhos dos sinais monitorados e do código da doença inócua d_λ, o endereço da população gerada

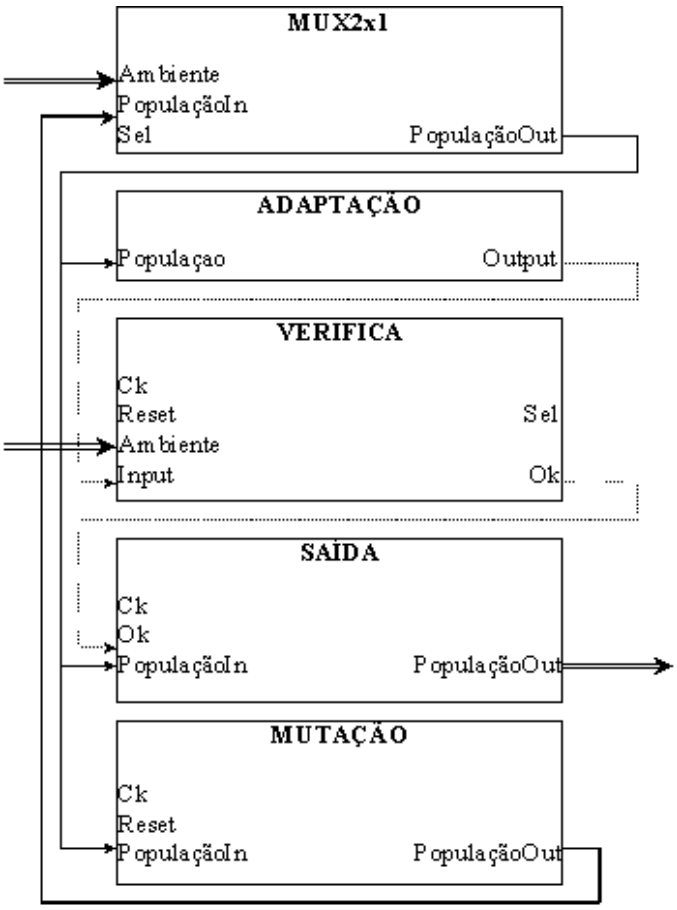


Figura 6.12: GAADT implementado na plataforma Chameleon

pelo bloco **MUTAÇÃO** e o sinal **Sel**. Se o sinal **Sel** estiver desativado, então o endereço da população atual **PopulaçãoIN** é passado para o endereço da **PopulaçãoOut**, senão o endereço da população vinda do **Ambiente** é passado para o endereço da **PopulaçãoOut**. O bloco **ADAPTAÇÃO** recebe o endereço da **População**, e armazena o valor da adaptação dos cromossomos em um lugar pré-determinado. Quando este bloco terminar sua tarefa ele ativa o sinal **Output**. O bloco **VERIFICAÇÃO** recebe o endereço do **Ambiente** e o sinal **Input** que é informada quando o cálculo da adaptação dos cromossomos da população foi concluído, e ativa ou desativa os sinais **Sel** e **Ok**, segundo as seguintes regras:

- se **Ambiente** for alterado e um critério de parada for atingido, então os sinais **Sel** e **Ok** são ativados.
- se **Ambiente** for alterado e até o momento nenhum critério de parada for atingido, então o sinal **Sel** é ativado e o sinal **Ok** é desativado;
- se **Ambiente** não for alterado e um critério de parada tiver sido atingido, então o sinal **Sel** é desativado e

o sinal **Ok** é ativado; e

- se **Ambiente** não for alterado e até o momento nenhum critério de parada for atingido, então os sinais **Sel** e **Ok** são desativados.

O bloco **Saída** retorna o endereço da população gerada pelo GAADT quando um dos critérios de adaptação for atingido, ativando neste mesmo instante o sinal **Ok**. O bloco **MUTAÇÃO** recebe a população atual e gera uma nova população formada pelos cromossomos da população atual e pelos cromossomos mutantes construídos a partir desta população.

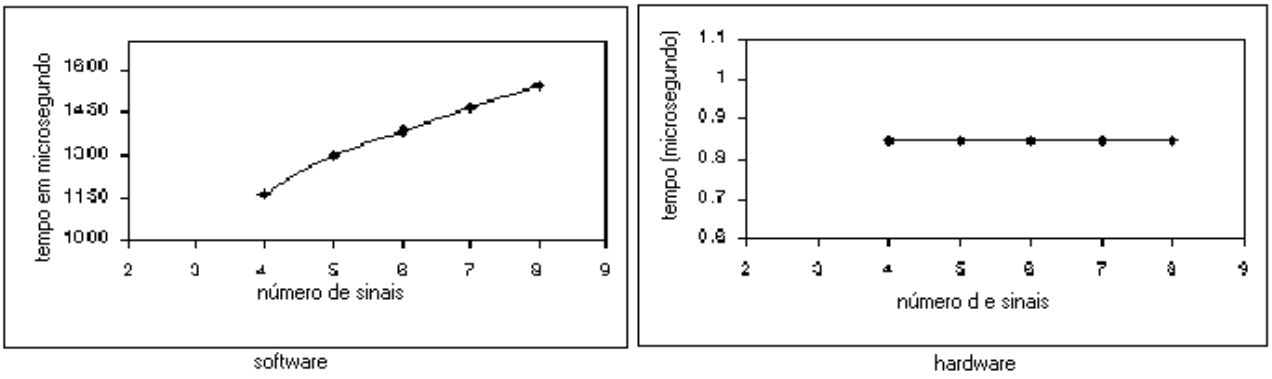


Figura 6.13: Tempo de Convergência

Para entender o comportamento do tempo de convergência do GAADT implementado em *hardware* e em *software*, variou-se o número de sinais monitorados entre 4 e 8, foram realizadas 100 execuções deste algoritmo considerando o critério de parada como sendo: encontrar um cromossomo com adaptação superior a 60%. O resultado está mostrado na Figura 6.13. A unidade de medida do tempo para o GAADT convergir nesta aplicação é o microsegundo.

6.6 Discussão

O equilíbrio apresentado pelo GAADT entre a adaptação média e o cromossomo mais adaptado versus o número de iteração pode ser explicado pelo comprometimento deste algoritmo com as características relevantes para o problema, refletida pela função gene dominante e grau de adaptação do gene, uma vez que o gene pertence a um nível de percepção epistemológica inferior ao cromossomo analisado pelos outros modelos de algoritmos genéticos considerados nestes estudos.

Deve-se observar ainda que entre os algoritmos genéticos investigados, o GAADT é o que requer o menor número de iterações para encontrar o cromossomo mais adaptado. A explicação para esto desempenho do GAADT deve-se ao número de características relevantes para o problema investigadas em cada iteração, que é o maior possível, uma vez que todos os cromossomos da população atual têm pelo menos uma oportunidade de gerar descendentes adaptados ao ambiente.

O comportamento desenhado nos gráficos da melhor adaptação encontrada em uma iteração do GAADT é linear porque a próxima população avaliada por este algoritmo tem sempre adaptação média maior ou igual à da população atual, já que a próxima população é composta pelos cromossomos da população atual que apresentam adaptação maior do que a adaptação média da população, adotado como critério pelo predicado p_{corte}; pelos cromossomos gerados por cruzamento, que por serem formados pelos genes dominantes dos cromossomos pais possuem na pior das hipóteses o mesmo valor de adaptação dos seus pais; e pelos cromossomos construídos pela mutação, que por definição têm adaptação maior que a dos cromossomos que lhes deram origem.

Analisando os gráficos da melhor adaptação encontrada em uma iteração da Seções 6.2, 6.3 e 6.4 verifica-se que o segundo algoritmo mais bem sucedido foi o algoritmo genético baseado em aprendizagem por *linkage*. Isto se justifica porque o algoritmo genético de tempo contínuo foi proposto com o intuito de garantir a diversidade da população e assim evitar a convergência prematura.

A superioridade e constância do tempo de convergência da implementação em *hardware* em comparação com a implementação em *software* se explica pelo paralelismo envolvido nas atividades de cálculo da função de adaptação, na construção de cromossomos mutantes, na geração da próxima população, conforme ilustrado pela arquitetura da Figura 6.12, e também pela maior velocidade do código implementado em FPGA.

Capítulo 7

Conclusão

7.1 Contribuições e Relevância

Neste trabalho foram apresentados um modelo de algoritmo genético baseado em tipos abstratos de dados, exemplos de aplicações do referido algoritmo com especificação e prova de refinamento em Z, e uma breve análise dos resultados obtidos pelo algoritmo proposto em comparação com os resultados obtidos pelo algoritmo genético de Holland e/ou por algumas das suas variações descritas no Capítulo 2.

O desenvolvimento do algoritmo genético baseado em tipos abstratos de dados obedeceu à idéia básica utilizada por Holland [79] de usar a metáfora da Teoria das Origens das Espécies de Charles Darwin. A diferença entre estes dois algoritmos reside principalmente na forma como o material genético manipulado é estruturado. Na estrutura proposta aqui, o cromossomo é um conjunto de genes que são n-uplas de bases, mais consonante com a construção genética do DNA dos seres vivos encontrada nos livros de Biologia Molecular do que a de Holland e seu seguidores, onde o cromossomo é um vetor definido sobre um dado alfabeto. Com esta percepção estruturada do cromossomo, a construção de algoritmos genéticos baseados em tipos abstratos de dados ganha maior flexibilidade de especificação. Além disso, o espectro de problemas para o qual este algoritmo pode ser adotado como mecanismo computacional de solução é maior, isto é, a família dos problemas que podem ser resolvidos por este mecanismo é mais abrangente. Tal ganho deve-se, principalmente, à separação da fase de análise de requisitos e especificação do ciclo de vida de um sistema, uma vez que a adequação do resultado do problema ao tipo cromossomo, no modelo de algoritmo genético baseado em tipos abstratos de dados, é metodologicamente adiada para a etapa posterior à análise do problema. Isto dá mais nitidez e sistemática ao modelo de algoritmo genético aqui proposto, por reduzir o esforço do usuário de tentar adequar o resultado de

um problema a uma estrutura de dados previamente definida, como postulado pelos demais modelos. No algoritmo genético baseado em tipos abstratos de dados, a estrutura de dados que será adotada para representar o resultado do problema depende da instanciação dos tipos base, gene e cromossomo especificados pelo usuário. Isto significa que a visão baseada em TAD adotada fornece uma moldura metodológica para o uso sistemático do GAADT.

O formalismo Z utilizado fez com que o desenvolvimento do GAADT se tornasse mais rigoroso na verificação de consistência e funcionalidade da construção adotada. Isto pode ser constatado pela especificação do GAADT apresentada no Capítulo 3 e das aplicações deste algoritmo apresentadas no Capítulo 5, bem como pela prova das propriedades exigidas para alguns elementos do GAADT mostradas no Apêndice C e pela prova de que este algoritmo é refinado pelas aplicações referidas e exibidas no Apêndice D. Assim, tal formalismo emprestou ao aparato teórico matemático usado na descrição do GAADT a garantia de operacionalidade necessária a um modelo computacional de uso geral idealizado para o algoritmo aqui proposto.

A adoção da idéia de Tipos Abstratos de Dados para a elaboração do algoritmo apresentou-se muito promissora no que diz respeito à sua qualidade ferramental. O ponto de vista a ser observado é o de que o GAADT não pretende simular o processo de evolução dos seres vivos observado na natureza, mas tão somente resolver problemas adotando o conceito de evolução para encontrar resultados de melhor qualidade construídos a partir de resultados de qualidade inferior.

O comprometimento da análise da intenção do problema na escolha dos objetos que devem figurar entre os ingrediente genéticos usados para a construção do cromossomo, metaforicamente correspondente ao DNA, certamente conduzirá à objetividade requerida na solução dos mesmos. Entre as vantagens operacionais apresentadas por este algoritmo estão:

- a liberação do uso de representações sugeridas pela análise do problema e, em conseqüência, a possibilidade de escolha de representações diferentes do tradicional vetor binário de tamanho fixo adotada pelos modelos de Holland e afins;
- o uso de uma representação para a população com tamanho variado e não limitado, impedindo a perda de cromossomos adaptados, gerados pela ação das operações de cruzamento e mutação, que podem ser descartados por exceder um tamanho pré-definido;
- a eliminação da operação de substituição, por ser epistemologicamente estranha à Teoria da Origem das

Espécies, onde o fator determinante da morte de um ser vivo é a sua baixa adaptação ao ambiente e não à manutenção do número de seres vivos na população;

- o comportamento das operações de cruzamento e a mutação, que varia em função da definição das funções e predicados utilizados na sua construção, e não em função da representação do cromossomo ou da natureza do problemas, como as operações descritas na Seção 2.2.5 e na Seção 2.2.6;

- a presença explícita do ambiente, como ingrediente influenciador e determinante da dinâmica evolutiva do comportamento das populações geradas pelo GAADT, permitindo o uso deste algoritmo no tratamento *on line* de problemas complexos em ambientes altamente variantes, como são os casos de: controle de trânsito de grandes cidade, acompanhamento de variações climáticas, ocorrência de secas, controle de abastecimento, etc.;

- o roteiro de especificação derivado da adoção de TAD, devidamente suportado pelo formalismo em Z, praticamente enseja uma metodologia para a solução de problemas cujas soluções são obtidas pelo uso do GAADT e os resultados obtidos pelo mesmo são os resultados desejados para o problema em questão;

- a adoção da adaptação média variante como parâmetro de seleção dos cromossomos que comporão a população seguinte, a cada passo do processo, simplifica a realização da intenção de resultados desejados em cada problema;

- o registro da relação de descendência entre os cromossomos das populações geradas durante o processo de evolução do GAADT permite a construção de uma explicação para o resultado encontrado, a qual sempre é requerida pelos problemas de aprendizagem;

- a definição da função peso de um gene no cromossomo permite modelar as relações de dependência entre o gene fornecido e os outros genes do cromossomo, garantindo assim a possibilidade de aplicação do GAADT a problemas com epistasis;

- o comportamento da operação de cruzamento garante a transmissão das características relevantes para o problema do cromossomo pai para o cromossomo filho, conduzindo assim a uma busca mais objetiva dos cromossomos promissores. Isto contribui para que a adaptação média da população seja estritamente monotônica;

- a aplicação da operação de mutação sobre os cromossomos não adaptados evita que os genes relevantes para o problema pertencentes a estes cromossomos sejam perdidos, além de garantir a diversidade da população.

Neste trabalho também foi apresentado um apanhado das tentativas de se estudar a convergência dos algoritmos de Holland e seus derivados, onde foi usada a teoria do *schema* [79], a hipótese de blocos de construção [58] e o teorema do ponto fixo de Banach [9]. No mesmo capítulo, foi desenvolvido um esboço de uma teoria para o estudo da convergência de processos evolutivos com o intuito de se estudar a convergência do GAADT e, posteriormente, dentro do mesmo aparato teórico, estudou-se a convergência deste algoritmo.

No estudo específico da convergência do GAADT ficou estabelecido que o mesmo converge independentemente da escolha da população inicial utilizada para a aplicação do processo e, para isto, foi exibida a prova de que o GAADT é um processo evolutivo monotônico, ver Seção 4.2. Esta convergência e as condições sob as quais a mesma se dá, mune o algoritmo aqui proposto de uma qualidade extraordinária do ponto de vista da segurança de aplicação. Em uma análise do GAADT objetivando a percepção da convergência depreende-se que o principal responsável por este fenômeno é a estratégia, fortemente objetiva, da adoção da adaptação média, variante a cada passo do processo, como parâmetro de validação dos cromossomos desejados como resultados satisfatórios do problema em observação. Tal estratégia, aliada à idéia da metáfora da evolução, dá ao GAADT esta qualidade de busca objetiva dos resultados desejados no problema.

7.2 Limitações e Restrições

Devido a limitação dos objetivos deste trabalho, ficaram fora do escopo vários estudos igualmente importantes para a consolidação do modelo proposto. Sem a idéia de exaurir estas questões, serão listadas abaixo algumas das mais importantes:

- como o cálculo da adaptação média pode ser eventualmente feito sobre uma população de grande tamanho, a convergência do GAADT pode se dar de modo lento e, em decorrência, a obtenção dos resultados satisfatórios do problema em questão pode demorar bastante. Isso acarreta a necessidade de, posteriormente, fazer-se um estudo sobre processos de aceleração do mesmo. Várias são as idéias a serem utilizadas como, por exemplo, a adaptação média ser obtida através de um processo de amostragem adequada, a cada passo, da população funcionalmente aplicada pelo algoritmo;

- considerando a peculiaridade da abordagem utilizada para a elaboração do GAADT, parece natural a procura de características dos problemas solúveis pelo mesmo, onde seriam levados em consideração os domínios do ambiente onde o problema se insere, a natureza da adaptação desejada, sugerida pela intenção do problema, as qualidades topológicas dos domínios acima citados, entre outras;

- um estudo das taxonomias das populações geradas pelo GAADT pode, eventualmente, tornar-se necessário para a preservação e conhecimento da história dos resultados obtidos pelo mesmo, isto é, um retrato dos caminhos percorridos pelo algoritmo para a obtenção dos resultados do problema;

- a inclusão de estratégias catalisadoras, para construção de um modelo derivado do GAADT, utilizando-se da metáfora de funcionamento das enzimas da biologia, para a inclusão de elementos de qualidade exógenos na solução de problemas oraculares, tais como problemas de natureza políticos-administrativos (planejamento urbano, catástrofes urbanas e regionais, etc.), problemas sociológicos (motivação de acentamentos populacionais, movimentos migratórios, consumo, etc.), tratamentos intensivos de paciente de UTI (onde o médico pode interferir com conhecimentos resultantes de sua experiência e intuição, mas não necessariamente constante da literatura médica catalogada, etc);

- em qualquer dos casos acima apresentados será necessário um estudo da complexidade do GAADT e seus eventuais derivados. Esta complexidade deve contemplar dois aspectos, pelo menos. O primeiro, é a complexidade do algoritmo quanto ao seu esforço computacional, e o segundo é quanto à sua natureza funcional, isto é, seu aspecto enquanto programa, sua complexidade estrutural (laços, aninhado, sua lógica, etc.).

A aplicação do GAADT, devido à sua forma de concepção, aponta para uma aplicabilidade muito ampla, entretanto, neste trabalho, ele foi aplicado ilustrativamente em alguns casos, representantes de paradigmas de classes de problemas, como é o caso do problema do caixeiro viajante, o problema de construção de árvores filogenéticas, o problema de controle de semáforos de trânsito e o problema do monitoramento de sinais vitais de um paciente de UTI. Em trabalhos futuros, o GAADT será utilizado para outra gama de problemas interessantes e importantes na consolidação da qualidade deste modelo.

Um outro aspecto que certamente contribuirá para tornar o GAADT um arcabouço de uso prático é o projeto e a implementação de um ambiente de desenvolvimento a partir da instanciação da especificação em Z

do GAADT. Tal ambiente deve estar conectado a um provador de teoremas como o Z/Eves, de forma a permitir o descarte mecânico das obrigações de prova conseqüentes da instanciação.

Bibliografia

[1] L. A. Albert, *Efficient genetic algorithms using discretion scheduling*, M.Sc. these, University of Illinois (2001).

[2] A. C. Albuquerque, *Uma lógica proposicional de resolução de problemas*, Dissertação de Mestrado, Departamento de Sistema e Computação, Universidade Federal da Paraíba (1998).

[3] D. S. Amorim, *Elementos básicos de sistemática filogenética*, Sociedade Brasileira de Entomologia de São Paulo (1975).

[4] H. J. Antonisse, *A new interpretation of schema notation that overturns the binary encoding constraint*, In J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989).

[5] ———, *A grammar-based genetic algorithm*, Proceeding of the Foundations of Genetic Algorithms (1991).

[6] J. Arabas, Z. Michalewics, and J. Mulawka, *GAVaPS - a genetic algorithm with varying population size*, Proceedings of the International Conference on Evolutionary Computation (1994).

[7] P. Bak and S. Boettcher, *Self-organized criticality and punctuated equilibrium*, Physica (1997).

[8] J. E. Baker, *Adaptive selection methods for genetic algorithms*, In J. J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms, pages 101-111, Lawrence Erlbaum Associates (1985).

[9] S. Banach, *Sur les opérations dans les ensembles abstraits et leur appliations aux équations intégrates*, Fundamenta Mathematica (1922).

[10] R. Basile and L. E. Magalhães, *Citologia e Genética*, CULTRIX - SP (1975).

[11] T. Bäck and H. P. Schwefwl, *Evolution strategies I: Variants and their computational implementation*, John Wiley e Sons (1995).

[12] T. C. Belding, *The distributed genetic algorithm revisited*, Proceedings of the International Conference on Genetic Algorithms (1995).

[13] R. K. Belew, *When both individuals and populations search: adding simple learning to the genetic algorithms*, In J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989).

[14] A. Bergman and M. W. Feldman, *Recombination dynamics and the fitness landscape*, Ph.D. these, University of Illinois (1992).

[15] N. Bernardi, *Resumo das aulas sobre: Sistemática Filogenética*, I Curso Especial de Sistemática Zoológica do Departamento de Ciências Biológicas, São Carlos - SP (1981).

[16] L. Booker, *Improving search in genetic algorithms*, In L. Davis, editor, Genetic Algorithms and Simulated Annealing, Pitman (1987).

[17] ———, *Reducing bias and inefficiency in the selection algorithm*, Proceedings of the International Conference on Genetic Algorithms (1987).

[18] J. C. Bowersox and R. L. Cornem, *Remote operative urology using a surgery telemanipulator system: preliminary observations*, Urology, MEDLINE (1998).

[19] J. C. Bowersox, A. Shah, J. Jensen, J. Hill, P. R. Cordts, and P. S. Green, *Vascular applications of telepresence surgery: initial feasibility studies in swine*, J. Vasc Surg, MEDLINE (1996).

[20] M. F. Bramlette, *Initialization, mutation and selection methods in genetic algorithms for function optimization*, In R. K. Belew and L. B. Booker, editor, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann (1991).

[21] A. Brindle, *Genetic algorithms for function optimization*, Ph.D. these, University of Alberta (1981).

[22] E. Cantú-Paz, *Designing efficient master-slave parallel genetic algorithms*, Illinois Genetic Algorithms Laboratory Report n^o 97004, University of Illinois (1997).

[23] ———, *A survey of parallel genetic algorithms*, Illinois Genetic Algorithms Laboratory Report n^o 97003, University of Illinois (1997).

[24] _____, *Designing efficient and accurate parallel genetic algorithms*, Illinois Genetic Algorithms Laboratory Report n^o 990017, University of Illinois (1999).

[25] _____, *Migration policies and takeover times in parallel genetic algorithms*, Illinois Genetic Algorithms Laboratory Report n^o 99008, University of Illinois (1999).

[26] _____, *Migration policies, selection pressure, and parallel evolutionary algorithms*, Illinois Genetic Algorithms Laboratory Report n^o 990015, University of Illinois (1999).

[27] _____, *Topologies, migration rates and multi-population parallel genetic algorithms*, Illinois Genetic Algorithms Laboratory Report n^o 99007, University of Illinois (1999).

[28] H. M. Cartwright and G. F. Mott, *Looking around: Using clues from the data space to guide genetic algorithm searches*, Proceedings of the International Conference on Genetic Algorithms (1989).

[29] Y. P. Chen and D. E. Goldberg, *Introducing start expression genes to the linkage learning*, Illinois Genetic Algorithms Laboratory Report n^o 2002007, University of Illinois (2002).

[30] C. R. Clement, *Descriptores minimos para el pejibaye (Bactris gasipaes h.b.k.) y sus implicaciones filogeneticas*, Dissertação de Mestrado, Departamento de Biologia, Universidade de Costa Rica (1986).

[31] R. P. B. G. Coutinho, R. V. Vieira, and M. E. Lima, *A genetic algorithm in hardware for vital signals monitoring*, Anais do Sbmicro2000 (2000).

[32] N. L. Cramer, *A representation for the adaptive generation of simple sequential programs*, In J. J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates (1985).

[33] E. Goodman D. Eby, R. Averill and W. Punch, *The optimization of flywheels using an injection island genetic algorithm*, Evolutionary Design by Computers (1999).

[34] C. Darwin, *A origem das espécies, trad. do original: "on the origin of species"*, Hemus Editora LTDA - SP.

[35] Y. Davidor, *Epistasis variance: Suitability of a representation to genetic algorithms*, Complex Systems (1990).

[36] L. Davis, *Job shop scheduling with genetic algorithms*, In J. J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates (1985).

[37] _____, *Adapting operator probabilities in genetic algorithms*, In J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989).

[38] _____, *Handbook of genetic algorithms*, Van Nostrand Reinhold - NY (1991).

[39] L. Davis and S. Coombs, *Genetic algorithms and communication link speed design: theoretical considerations*, In J. J. Grefenstette, editor, Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates (1987).

[40] D. C. Dennett, *A perigosa idéia de darwin: a evolução e os significados da vida*, Ciência Atual, Rocco (1995).

[41] Goldberg D. E., *A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing*, Complex Systems (1990).

[42] R. V. Vieira e M. A. Lopes, *An application of abstract data types in the genetic algorithms construction: The example of the traveling salesman problem*, Proceeding of the International Conference of the Chilean Computer Science Society (1999).

[43] _____, *Uma abordagem teórica inicial para os algoritmos genéticos através de tipos abstratos de dados*, Relatório Técnico n^o 002, Departamento de Informática, Universidade Federal de Pernambuco (1999).

[44] _____, *Alinhamento de dna´s via algorimo genético*, Encontro Nacional de Inteligência Artificial, Sociedade Brasileira de Computação (2000).

[45] _____, *An application of abstract data types in the genetic algorithms construction; the exemple of the traffic light control problem*, III CITS-IEEE (2000).

[46] _____, *A construction of phylogenetic trees by genetic algorithms*, Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biology Science (2000).

[47] L. J. Eshelman, *The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination*, Proceedings of the Foundations of Genetic Algorithms, Morgan Kaufmann (1991).

[48] L. J. Eshelman and J. D. Schaffer, *Real-code genetic algorithms and interval-schemata*, Proceedings of the Foundation of Genetic Algorithms (1993).

[49] L. K. Evans, *Embracing premature convergence*, Proceedings of the International Conference on Evolutionary Computation, Morgan Kaufmann (1998).

[50] P. Field, *A multary theory for genetic algorithms: Unifying binary and nonbinary problem representations*, Ph.D. these, University of Lodon (2000).

[51] B. R. Fox and M. B. McMahon, *Genetic operators for sequencing problems*, Proceedings of the Foundations of Genetic Algorithms, Morgan Kaufmann (1991).

[52] D. J. Futuyma, *Biologia evolutiva*, Sociedade Brasileira de Genética/CNPq (1992).

[53] F. Ghannadian, R. Shonkwiler, and C. O. Alford, *Performance evaluation of iterative improvement algorithms combined with random restart applied to scheduling problems*, Proceedings of the International Conference of Information Science: Fuzzy Logic, Intelligent Control e Genetic Algorithm (1997).

[54] Steven Gold, *Comparison of global optimization methods*, Project Presentation for ME 761, University of Massachusetts (1995).

[55] D. E. Goldberg, *The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity*, Evolutionary Design by Computer, YEAR =.

[56] ———, *Optimal initial population size for binary-coded genetic algorithms*, Report n^o 85001, University of Alabama (1985).

[57] ———, *Simple genetic algorithms and the minimal, deceptive problem*, Genetic Algorithms and Simulated Annealing, Pitman (1987).

[58] ———, *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley (1989).

[59] ———, *An analysis of boltzmann tournament selection: Part ii: An experimental analysis of boltzmann tournament selection*, Illinois Genetic Algorithms Laboratory Report n^o 94007, University of Illinois (1994).

[60] ———, *Genetic algotithms in search, optimization, and machine learning*, Addison Wesley Massachusetts (1997).

[61] ———, *Discovering deep building blocks for competent genetic algorithms using change discovery via key graphs*, Illinois Genetic Algorithms Laboratory Report n^o 2002026, University of Illinois (2002).

[62] D. E. Goldberg, B. Korb, and K. Deb, *Messy genetic algorithm: Motivation, analysis, and first results*, Complex Systems (1989).

[63] D. E. Goldberg and R. Lingle, *Alleles, loci and the TSP*, In J. J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates (1985).

[64] D. E. Goldberg and L. Wang, *Adaptive niching via coevolutionary sharing*, Illinois Genetic Algorithms Laboratory Report n^o 97007, University of Illinois (1997).

[65] D.E. Goldberg and K. Deb, *A comparative analysis of selection schemes used in genetic algorithms*, In G. J. E. Rawlins, editor, Proceedings of the Foundations of Genetic Algorithms, Morgan Kaufmann (1991).

[66] S. J. Gould, *Darwin e os grandes enigmas da vida*, Coleção Ciência Aberta, Martins Fontes - SP (1992).

[67] P. Graham and B. Nelson, *Genetic algorithms in software and in hardware - a performance analysis of workstation and custom computing machine implementations*, Symposium on FPGAs for Custom Computing Machines (1996).

[68] J. J. Grefenstette, *Optimization of control parameters for genetic algorithms*, IEEE Transactions on Systems, Man, and Cybernetics, vl. 16, YEAR =.

[69] _____, *Deception considered harmfun*, Proceedings of the Foundations of Genetic Algorithms (1992).

[70] J. J. Grefenstette, R. Gobal, B. Rosmaita, and D. Van Gucht, *Genetic algorithm for the TSP*, Proceedings of the International Conference on Genetic Algorithms (1985).

[71] J. L. Grosfeld, *Visions: medical education and surgical training in evolution*, Arch Surg, MEDLINE (1999).

[72] L. Groves, *Refinement and the Z Schema Calculus*, An FME sponsored Refinement Workshop in collaboration with BCS FACS, Proceedings of the FME (2002).

[73] G. R. Harik, *Finding multimodal solutions using restricted tournament selection*, Proceedings of the International Conference on Genetic Algorithms (1995).

[74] _____, *Learning gene linkage to efficiently solve problems of bound difficulty using genetic algorithms*, Ph.D. these, University of Michigan (1997).

[75] G. R. Harik, E. Cantú-Paz, D. E. Goldberg, and L. B. Miller, *The gambler's ruin problem. genetic algorithms and the sizing of population*, Illinois Genetic Algorithms Laboratory Report n^o 96004, University of Illinois (1996).

[76] _____, *The gambler's ruin problem, genetic algorithms, and the sizing of populations*, Proceedings of the International Conference on Evolution Computation (1997).

[77] R. Hauser and R. Männer, *Implementation of standard genetic algorithm on MIND machines*, Parallel Problem Solving from Nature III (1994).

[78] D. Heckerman, D. Geiger, and M. Chickering, *Learning bayesian networks: The combination of knowledge and statistical data*, Illinois Genetic Algorithms Laboratory Report n^o 94009, University of Illinois (1994).

[79] J. H. Holland, *Adaptation in Natural and Artificial Systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press (1975).

[80] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. Thagard, *Induction: Processes of inference, learning, and discovery*, MIT Press (1986).

[81] A. Homaifar, S. H. Y. Lai, and X. Qi, *Constrained optimization via genetic algorithms*, Simulation (1994).

[82] C. V. Hoyweghen, D. E. Goldberg, and B. Naudts, *Building block superiority multimodality and synchronization problems*, Illinois Genetic Algorithms Laboratory Report n^o 2001020, University of Illinois (2001).

[83] C. Hsu, H. Takahashi, K. Shida, H. Fujikawa, and S. Yamada, *An eugenic mutations for optimum problems*, IEEE Press (1995).

[84] I. Hys, *Specification case studies*, Prentice-Hall International (1993).

[85] C. Z. Janikow and Z. Michalewicz, *An experimental comparison of binary an floating point representations in genetic algorithms*, Proceedings of the International Conference on Genetic Algorithms (1991).

[86] P. Jog, J. Y. Suh, and D. V. Gucht, *The effects of population size, heuristic crossover, and local improvement on a genetic algorithm for the traveling salesman problem*, Proceedings of the International Conference on Genetic Algorithms (1989).

[87] M. E. Johnston, K. B. Langton, R. B. Haynes, and A. Mathieu, *Effects of computer-based clinical decision support system on clinician performance and patient outcome: a critical appraisal of research*, Ann Intern Med, MEDLINE (1994).

[88] J. A. Joines and C. R. Houck, *On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs*, Proceedings of the International Conference on Evolutionary Computation (1994).

[89] K. A. De Jong, *An analysis of the behavior of a class of genetic adaptive systems*, Ph.D. These, University of Michigan (1975).

[90] ———, *Genetic algorithms: A 10 year perspective*, Proceedings of the International Conference on Genetic Algorithms (1985).

[91] ———, *Learning with Genetic Algorithms: an overview*, Machine Learning (1988).

[92] ———, *Genetic algorithms are not function optimizers*, Foundations of Genetic Algorithms (1992).

[93] K. A. De Jong, M. A. Potter, and W. M. Spears, *Using problem generators to explore the effects of epistasis*, Proceedings of the International Conference on Genetic Algorithms (1997).

[94] K. A. De Jong and W. M. Spears, *Using genetic algorithms to solve NP-complete problems*, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989).

[95] H. Kargupta, *The Gene Expression Messy Genetic Algorithm*, Proceedings of the International Conference on Evolutionary Computation (1996).

[96] N. Khan, D. E. Goldberg, and M. Pelikan, *Multi-objective bayesian optimization algorithm*, Illinois Genetic Algorithms Laboratory Report n^o 2002009, University of Illinois (2002).

[97] R. Kikinis, L. P. Gleason, and F. A. Jolesz, *Surgical planning using computer-assisted three-dimensional reconstruction*, In: R.H. Taylor, S. Lavallee , G. C. Burdea e R. Moesges, Computer-Integrated Surgery: Technology and Clinical Applications, Cambridge, Mass: MIT Press (1996).

[98] M. C. King and A. C. Wilson, *Evolution at two levels in humans and chimpanzees*, Science (1975).

[99] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, Science (1983).

[100] D. Knjazew, *Application of the fast messy genetic algorithm to permutation and scheduling problems*, Illinois Genetic Algorithms Laboratory Report n^o 2000022, University of Illinois, (2000).

[101] D. Knjazew and D. E. Goldberg, *Large-scale permutation optmization with the ordering messy genetic algorithm*, Illinois Genetic Algorithms Laboratory Report n^o 2000012, University of Illinois (2000).

[102] T. Koyama, H. Okudera, H. Gibo, and S. Kobayashi, *Computer-generated microsurgical anatomy of the basilar artery bifurcation: technical note*, Neurosurg, MEDLINE (1999).

[103] J. R. Koza, *Genetic Programming: A paradigm for genetically breeding populations of computer programs to solve problems*, University of Stanford, Computer Science Department, technical report STAN-CS-90-1314 (1990).

[104] _____, *A hierarchical approach to learning the boolean multiplexed function*, Proceedings of the Foundations of Genetic Algorithms (1991).

[105] _____, *Genetic programming: On the programming of computers by means of natural selection*, Cambridge, MA: MIT Press (1992).

[106] A. J. Owens L. J. Fogel and M. J. Walsh, *Artificial intelligence through simulated evolution*, Jhon Whiley (1966).

[107] G. H. M. Lawrence, *Taxonomy of vascular plants*, Macmillan (1951).

[108] M. E. Lima, S. V. Cavalcante, C. C. Araújo, and H. B. S. Leão, *Chamaleon: A prototyping plataform for digital system design*, Proceedings I, XIII SBMicro (1998).

[109] F. G. Lobo, K. Deb, D. E. Goldberg, G. R. Harik, and L. Wang, *Compressed intros in a linkage learning genetic algorithm*, Illinois Genetic Algorithms Laboratory Report n^o 97010, University of Illinois (1997).

[110] F. G. Lobo and D. E. Goldberg, *The parameter-less genetic algorithm in practice*, Illinois Genetic Algorithms Laboratory Report n^o 2001022, University of Illinois (2001).

[111] M. A. Lopes, *Introdução a uma teoria geral de problemas*, Tese de Doutorado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (1981).

[112] S. W. Mahfoud, *Crowding and preselection revisited*, Proceedings of the Parallel problem solving from nature (1992).

[113] _____, *Population sizing for sharing methods*, Illinois Genetic Algorithms Laboratory Report n^o 94005, University of Illinois (1994).

[114] B. Malott, R. C. Averill, E. D. Goodman, Y. Ding, and W. F. Punch, *Use of genetic algorithms for optimal design of laminated composite sandwich panel with bending-twisting coupling*, AIAA SDM (1996).

[115] J. Maresky, Y. Davidor, D. Gitler, G. Aharoni, and A. Barak, *Selectively destructive re-start*, Proceedings of the International Conference on Genetic Algorithms (1995).

[116] I. Meisels, *Software Manual for Windows Z/EVES Version 1.5 and the Z Browser*, (1997).

[117] I. Meisels and M. Saaltink, *The Z/Eves Reference Manual*, ORA Canada (1997).

[118] H. Mühlenbein, *How genetic algorithms really work I: Mutation and hillclimbing*, Parallel problem solving from nature (1992).

[119] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*, Springer-Verlag (1996).

[120] B. L. Miller and M. J. Shaw, *Genetic algorithms with dynamic niche sharing for multimodal function optimization*, Proceedings of the International Conference on Evolutionary Computation (1996).

[121] M. Pelikan, *Bayesian optimization algorithms: From single level to hierarchy*, M.Sc. these, University of Illinois (2002).

[122] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, *Linkage problem, distribution estimation, and bayesian network*, Illinois Genetic Algorithms Laboratory Report n^o 98013, University of Illinois (1998).

[123] M. Pelikan, D. E. Goldberg, and K. Sastry, *Bayesian optimization algorithm, decision graphs, and occam´s razor*, Illinois Genetic Algorithms Laboratory Report n^o 2000020, University of Illinois (2000).

[124] M. Pelikan, D. E. Goldberg, and S. Tsutsui, *Combining the strengths of the bayesian optimization algorithm and adaptive evolution strategies*, Illinois Genetic Algorithms Laboratory Report n^o 2001023, University of Illinois (2001).

[125] A. Pommert, M. Riemer, T. Schiemenn, R. Schubert, U. Tiede, and K. H. Hoehne, *Three-dimensional imaging in medicine: methods and applications*, In: Taylor R.H., Lavallee S., Burdea G.C. and Moesges R., Computer-Integrated Surgery: Technology and Clinical Applications, MIT Press (1996).

[126] B. Potter, J. Sinclair, and D. Till, *An Introduction to Formal Specification and Z*, Prentice Hall (1991).

[127] D. Powell and M. M. Skolnick, *Using genetic algorithms in engineering design optimization with nonlinear constraints*, Proceedings of the International Conference on Genetic Algorithms (1993).

[128] T. A. Pryor, *Development of decision support systems*, Clin Monitoring Computing (1990).

[129] W. F. Punch, R. C. Averll, E. D. Goodman, S. C. Lin, and Y. Ding, *Design using genetic algorithms - some results for laminated composite structure*, IEEE Expert (1995).

[130] W. F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. Houland, and R. Enbody, *Further research on feature selection and classification using genetic algorithms*, Proceedings of the International Conference on Genetic Algorithms (1993).

[131] I. Rechenberg, *Evoluionstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution.*, Frommann-Holzboog (1973).

[132] C. R. Reeves, *Using genetic algorithms with small populations*, Proceedings of the International Conference on Genetic Algorithms (1993).

[133] J. T. Richardson, M. R. Hilliard, M. R. Palmer, and G. E. Liepins, *Some guidelines for genetic algorithms with penalty functions*, In J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989).

[134] T. Rintala, *Hardware implementation of genetic algorithms*, www.uwasa.fi.cs.publications/2NWGA/node60.html (1996).

[135] C. T. Rizzini, *Trabalho de fitogeografia do brasil*, Hueitec-Edu (1979).

[136] S. Roman, *Field theory*, Springer-Verlag (1995).

[137] F. Rothlauf, *The influence of binary representation of integers on the performance of selector combinative genetic algorithms*, Illinois Genetic Algorithms Laboratory Report n^o 2002014, University of Illinois (2002).

[138] F. Rothlauf, D. E. Goldberg, and A. Heinzl, *Bad coding and the utility of well-designed genetic algorithms*, Illinois Genetic Algorithms Laboratory Report n^o 2000007, University of Illinois (2000).

[139] S. M. Sait and H. Youssef, *VLSI physical design automation*, Paperback (1995).

[140] K. Sastry, *Evaluation relaxation schemes for genetic and evolutionary algorithms*, M.Sc. these, University de Illinois (2001).

[141] K. Sastry and D. Goldberg, *Modeling tournament selection with replacement using apparent added noise*, Illinois Genetic Algorithms Laboratory Report n^o 2001014, University of Illinois (2001).

[142] K. Sastry and D. E. Goldberg, *On extended compact genetic algorithm*, Illinois Genetic Algorithms Laboratory Report n^o 2000026, University of Illinois (2000).

[143] ———, *Genetic algorithm, efficiency enhancement, and deciding well with differing fitness bias values*, Illinois Genetic Algorithms Laboratory Report n^o 2002003, University of Illinois (2002).

[144] ———, *Genetic algorithm, efficiency enhancement, and deciding well with differing fitness variances*, Illinois Genetic Algorithms Laboratory Report n^o 2002002, University of Illinois (2002).

[145] ———, *How well does a single-point crossover mix building blocks with tight linkage?*, Illinois Genetic Algorithms Laboratory Report n^o 2002013, University of Illinois (2002).

[146] J. D. Schaffer, *Learning multiclass pattern discrimination*, Proceedings of the International Conference on Genetic Algorithms (1985).

[147] A. D. Scott, *HGA: A hardware-based genetic algorithm*, M.Sc. these, University of Nebraska (1994).

[148] H. Shimodai, *A diversity control oriented genetic algorithm (DCGA): Development and experimental results*, Proceedings of the Genetic and Evolutionary Computation Conference (1999).

[149] ———, *A diversity control oriented genetic algorithm (DCGA): Performance in function optimization*, Proceedings of the Genetic and Evolutionary Computation Conference (2000).

[150] R. E. Smith, *Adaptively resizing population: An algorithm and analysis*, Proceedings of the International Conference on Genetic Algorithms (1993).

[151] S. F. Smith, *A learning system based on genetic adaptive*, Ph.D. these, Science Computer, University of Pittsburgh (1980).

[152] W. M. Spears and K. A. De Jong, *An analysis of multi-point crossover*, Proceedings of the Foundation of Genetic Algorithms.

[153] J. M. Spivey, *The Z notation*, Prentice-Hall International (1992).

[154] R. P. Srivastava, *Time continuation in genetic algorithms*, M.Sc., University de Illinois (2001).

[155] J. Stoy, *Denotational semantics: the scott-strachey approach to programming language theory.*, MIT Press (1977).

[156] G. Syswerda, *Uniform crossover in genetic algorithms*, In J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989).

[157] A. Szalas and Z. Michalewics, *Contractive mapping genetic algorithms and their convergence*, Report n^o 96014, University of North Carolina (1993).

[158] D. M. Tade and R. E. Smith, *Expected allele coverage and the role of mutation in genetic algorithms*, Proceedings of the International Conference on Genetic Algorithms (1989).

[159] K. S. Tag, K. F. Man, S. Kwong, and Q. He, *Genetic algorithms and their applications*, Signal Processing Magazine (1996).

[160] S. Tsutsui and Y. Fujimoto, *Forking genetic algorithm with blocking and shrinking modes (fGA)*, Proceedings of the International Conference in Genetic Algorithms (1993).

[161] M. Vose and G. Liepins, *Punctuated equilibrium in genetic search*, Complex Systems (1991).

[162] ———, *Schema disruption*, Proceedings of the International Conference on Genetic Algorithms (1991).

[163] D. Whitle, *The GENITOR algorithms and selection pressure: why rank-based allocation of a reproductive trials is best*, Proceedings of the International Conference on Genetic Algorithms (1989).

[164] D. Whitle and T. Starkweather, *GENITOR II : A distributed genetic algorithm*, Journal of Experimental and Theoretical Artificial Intelligence (1990).

[165] J. R. Wilcox, *Organizational learning within a learning classifier system*, M.Sc. these, University of Illinois (1995).

[166] E. O. Wiley, D. Siegel-Causey, D. R. Brooks, and V. A. Funk, *The compleast cladist: A primer of philogenetic procedures*, Publicação especial n^o 19, Museu de História Natural, University of Kansas (1991).

[167] D. J. Winter, *The structure of field*, Springer-Verlag (1974).

[168] D. Yuret, *From genetic algorithms to efficient optimization*, M. Sc. these, University of Massachusetts (1994).

Apêndice A

A linguagem de especificação formal Z

A.1 Introdução

A linguagem Z é baseada na teoria de conjuntos e na lógica de predicados de primeira ordem [117, 116]. Ela foi desenvolvida pelo Programming Research Group (PRG) no laboratório de computação da Universidade de Oxford e em outros locais desde o final da década de 70, inspirada pelo trabalho inicial de Jean-Raymond Abrial. Este formalismo suporta o processo de desenvolvimento de sistemas de computador através do uso de vários cálculos de refinamento.

A utilização da linguagem Z é aconselhada para especificações abstratas de sistemas de computador para garantir que os elementos dos conjuntos definidos na especificação tenham a mesma natureza e para evitar paradoxos que podem conduzir à construção de conjuntos contraditórias. A linguagem Z lida com estes problemas, impondo uma disciplina na construção de objetos e expressões. Esta disciplina está baseada na noção de tipos, já que os objetos do sistema devem pertencer a um único tipo abstrato de dados e as expressões devem receberem termos de tipos adequados.

Nas seções que seguem são mostrados alguns dos componentes da notação Z utilizados no desenvolvimentos das especificações e provas apresentados nesta tese.

A.2 Tipos

Um tipo é um conjunto de dados, ou valores, o qual é associado a um objeto de maneira inalterável ao longo da especificação. A linguagem Z trabalha com tipos simples (como *given set*, inteiro, natural, etc.) e compostos (como conjunto, seqüência, produto cartesiano, etc.). Os objetos de um certo tipo só poderá participar de expressões ou predicados descritos em função de operadores que trabalham sobre este tipo.

A.2.1 Tipo Simples

Os *given sets* são tipos pré-definidos pelo usuário cuja estrutura interna não é detalhada. A única exigência apresentada para estes tipos é que eles devem ser disjuntos dois a dois. A declaração de um *given set* simplesmente contém o seu nome entre os símbolos "[" e "]". O nome de um *given set* é usado no texto da especificação para designá-lo. A seguir será apresentado a declaração dos tipos pessoa e data.

$$[PESSOAS, DATAS]$$

Outro tipo também definido pelo usuário oferecido é o tipo enumerado, que compreende uma coleção de objetos associados por alguma propriedade. A declaração de um tipo enumerado começa com o seu nome, seguido dos símbolos "::=", seguido da lista de objetos associados separados pelo símbolo "|". Por exemplo, a declaração dos meses do ano seria:

$$Meses ::= Janeiro \mid Fevereiro \mid Maro \mid Abril \mid Maio \mid Junho \mid Julho \mid Agosto \mid Setembro \mid Outubro \mid Novembro \mid Dezembro$$

Além dos tipos simples definidos pelo usuário, a linguagem Z dispões do tipo número naturais e inteiros que fazem parte da sua definição, denotados por $\mathbb{N}$ e $\mathbb{Z}$ respectivamente. Os operadores associados a estes tipos são de natureza aritmética e relacional tradicionais, os quais são designados pelos mesmos símbolos usados na matemática para representá-los.

A.2.2 Tipo Composto

O tipo conjunto pode ser declarado pela lista dos seus componentes separados pelo símbolo "," delimitada pelos símbolos "{" e "}" ou pela definição de uma variável associada a um tipo já definido, após vem o símbolo "|", seguido pela descrição das características comuns aos objetos do conjunto tudo delimitado pelos símbolos "{" e "}". As características compartilhadas pelos objetos são declaradas por predicados ou expressões matemática, os predicados serão detalhados mais tarde. Por exemplo, o conjunto dos números naturais menor do que cinco, $\{0, 1, 2, 3, 4\}$ ou $\{n : \mathbb{N} \mid n < 5\}$.

Um conjunto em Z pode ter mais de um nome na mesma especificação, para tanto basta escreve o novo nome do conjunto, em seguida o símbolo "$==$" e depois o nome atual do conjunto. Por exemplo, se o conjunto

$MenorCinco == \{0, 1, 2, 3, 4\}$ já tiver sido declarado, então depo-se declarar o conjunto $MenorIgualQuatro == MenorCinco$. O conjunto vazio de qualquer conjunto é representado pelo símbolo $\emptyset$.

Um conjunto particular oferecido é o subintervalo de números inteiros. A declaração deste conjunto obedece a seguinte sintaxe escreve-se os números que delimitam o intervalo em ordem crescente separados pelo símbolo "...". Por exemplo, o conjunto $\{0, 1, 2, 3, 4\}$ poderia ter sido declarado assim $0 \ldots 4$. A Tabela A.1 apresenta as operações disponíveis sobre o tipo conjunto.

operação	descrição
$\mathbb{P}X$	conjunto das partes do conjunto X
$X \subseteq Y$	conjunto X é um subconjunto próprio do conjunto Y
$X \subset Y$	conjunto X é um subconjunto do conjunto Y
$X \cup Y$	a união do conjunto X com o conjunto Y
$X \cap Y$	a interseção do conjunto X com o conjunto Y
$X \setminus Y$	a diferença do conjunto X com o conjunto Y
$\bigcup X$	a união dos conjuntos que fazem parte do conjunto X
$\bigcap X$	a interseção dos conjuntos que fazem parte do conjunto X

Tabela A.1: Operações definidas sobre o tipo conjunto

O tipo n-upla $(x_1, x_2, ..., x_n)$ é declarado obedecendo a seguinte regra: escreve-se o nome dos tipos associados as variáveis x_w, com $1 \leq w \leq n$, na ordem em que estas ocorrem, todos separados pelos símbolo $\times$. Por exemplo, o produto cartesiano dos conjuntos $Binario == \{0, 1\}$ e $Vogais == \{a, , e, i, o, u\}$ em Z seria declarado da seguinte forma $ProdutoCartesiano : Binario \times Vogais$, que corresponderia ao seguinte conjunto $\{(0, a), (0, e), (0, i), (0, o), (0, u), (1, a), (1, e), (1, i), (1, o), (1, u)\}$.

O tipo relação é declarado de acordo com a seguinte sintaxe: o nome do conjunto fonte, seguido pelo símbolo "$\leftrightarrow$", que é seguido pelo nome do conjunto imagem. Por exemplo, a definição da relação com conjunto fonte X e imagem Y é $X \leftrightarrow Y$. A Tabela A.2 apresenta algumas das operações disponíveis sobre o tipo relação.

operação	descrição
dom R	retorna o conjunto fonte da relação R
ran R	retorna o conjunto imagem da relação R
id R	retorna a relação identidade de R
$Q \ {}^{0}_{9} R$	retorna a relação formada pela composição das relações Q e R

Tabela A.2: Operações definidas sobre o tipo relação

O tipo função é declarado como: o nome do conjunto fonte, seguido pelo símbolo "$\rightarrow$", que é seguido pelo

nome do conjunto imagem. Por exemplo, a definição da função com conjunto fonte $\mathbb{N}$ e imagem $Meses$ é $\mathbb{N} \pfun Meses$.

O tipo seqüência é declarado assim: símbolo " seq " antecede o nome do tipo dos elementos que compõe a seqüência. Por exemplo, uma sequencia de números inteiros é declarada como $\mathrm{seq}\ \mathbb{N}$. A seqüência vazia é representada pelo símbolo $\langle\rangle$. A Tabela A.3 apresenta algumas das operações disponíveis sobre o tipo seqüência.

operação	descrição
$s \frown t$	concatena a seqüência s com a seqüência t
$head\ (t)$	retorna o elemento na primeira posição da seqüência t
$last\ (t)$	retorna o último na primeira posição da seqüência t
$tail\ (t)$	retorna a seqüência t sem o elemento da primeira posição
$front\ (t)$	retorna a seqüência t sem o elemento da última posição

Tabela A.3: Operações definidas sobre o tipo seqüência

A.3 Variável

A declaração de uma variável só pode ser feita após a especificação do seu tipo. Variáveis são declaradas construindo uma lista de seus nomes, separados por ",", seguida pelo símbolo ":", após o que vem o nome do tipo que será associado às variáveis da lista. Por exemplo, a declaração de uma variável p do tipo pessoa e de duas variáveis (d_1 e d_2) do tipo data são apresentadas a seguir:

$$\left|\begin{array}{l} p : PESSOAS \\ d_1, d_2 : DATAS \end{array}\right.$$

É possível também escrever estas duas declarações na mesma linha, desde que um símbolo "; " seja colocado entre elas, da seguinte forma:

$$\left|\begin{array}{l} p : PESSOAS;\ d_1, d_2 : DATAS \end{array}\right.$$

Por convenção quando uma variável é lida seu nome recebe o sufixo "?", e quando ela é escrita o sufixo "!". As únicas variáveis que podem mudar de valor são as variáveis de estado. A mudança de valor de uma variável

de estado é indica pelo símbolo "′". Por exemplo, dada uma variável de estado $n : \mathbb{N}$, para incrementar seu valor escreve-se: $n' = n + 1$, lê-se o novo valor da variável de estado n' é igual ao seu valor atual n acrescido de 1.

A.4 Predicado e Expressão

A declaração de predicados envolve um conjunto de variáveis e operadores (aritmético e relacionais) que trabalham sobre o mesmo tipo. Por exemplo, se uma especificação trabalha com dois tipos distintos $TIPO_1$ e $TIPO_2$, então esta especificação em princípio deveria apresentar dois símbolos distintos para a relação de igualdade, uma para os elementos do tipo $TIPO_1$ e outra para os elementos do tipo $TIPO_2$.

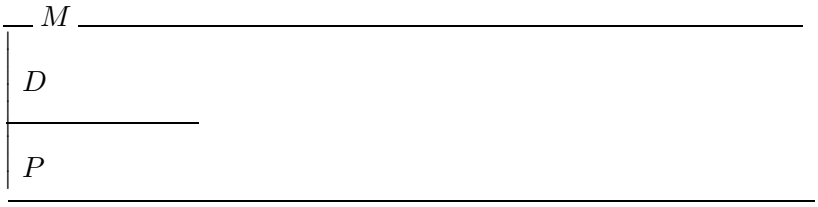
A notação Z nos permite criar predicados que obedecem a uma sintaxe parecia com a utilizada para descrever os predicados da lógica de primeira ordem. Por exemplo, se $TIPO$ é um tipo declarado em Z, então $x, y : TIPO$ é uma declaração de duas variáveis associadas a este tipo. Por conseguinte, o predicado $x, y : TIPO \bullet (x \neq y)$ da lógica seria escrito em Z como $x, y : TIPO \bullet (x \neq y)$, o símbolo "•" é usado em Z para separar a declaração das variáveis manipuladas pelos termos lógicos de um predicado de primeira ordem. Embora a sintaxe destes predicados seja ligeiramente diferentes, elas especificam a mesma semântica. As variáveis envolvidas na construção de um predicado devem estar previamente declaradas.

A notação Z permite que se trabalhe com predicados quantificadores existencial e universal. Um predicado quantificado existencialmente obedece a seguinte forma $D \mid R \bullet P$, lê-se existem variáveis declaradas em D, que satisfazem o predicado R as quais também satisfazem o predicado P. De maneira similar, um predicado quantificado universalmente é da forma $D \mid R \bullet P$, lê-se todas as variáveis declaradas em D, que satisfazem o predicado R também satisfazem o predicado P. As variáveis declaradas em D podem ocorrer livremente em R e P e as variáveis que não são declaradas em D devem constar de uma declaração prévia. O predicado R pode ser omitido sempre que desejado.

A.5 Esquemas

O estado e as operaçõe de um sistema especificado em Z são modelados através de uma estrutura própria denominada de esquemas. Um esquema é uma estrutura composta de duas partes, uma declarativa e outra

predicativa.



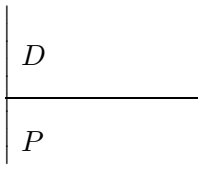
onde M é o nome associado ao esquema, D é sua parte declarativa e P é sua parte predicativa.

A parte declarativa de um esquema deve conter a especificação de todas as variáveis que ocorrem na parte predicativa. Nada impede que a parte declarativa contenha a especificação de outras variáveis que não ocorrem na parte predicativa.

A parte predicativa, por sua vez, deve listar todas as relações, ou ações, que se deseja realizar entre as variáveis de interesse. São estas ações que caracterizam o efeito esperado quando se chama o esquema especificado.

A.6 Definições Axiomáticas

As constantes de um sistema especificado em Z são modelados através de uma estrutura própria denominada de definições axiomáticas. Uma definição axiomática é uma estrutura composta de duas partes, uma declarativa e outra predicativa assim como o esquema.



Apêndice B

FPGA

B.1 Introdução

O FPGA é um dispositivo eletrônico VLSI (*Very Large Scale Integration* com uma alta capacidade lógica [139], que pode ser reprogramado diversas vezes, adaptando-se às diferentes aplicações do usuário. Por isso o FPGA é considerado o dispositivo ideal para a prototipação de sistemas digitais e uma solução para sistemas dinâmicos. A arquitetura de FPGA encontradas no mercado variando de acordo com o fabricante.

As arquiteturas de FPGA da Xilinx caracterizam-se por serem flexíveis e reprogramáveis. Estes dispositovos são compostos por blocos de construção lógica (CLB) que fornecem elementos funcionais para a construção da lógica do usuário; por blocos de entrada e saída (IOB) que mapeiam a interface entre os pinos externos e as linhas de sinais internos; e por interconexões que são meios de rotear a conecção entre as entradas e saídas dos CLB e IOB (ver Figura B.1).

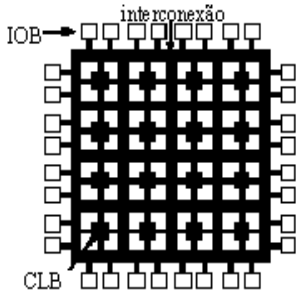


Figura B.1: Arquitetura do dispositivo SRAM da Xilinx

A ferramenta disponibilizada pela Xilinx para programação de componentes reconfiguráveis cobre aspectos do projeto esquemático, comportamental, *floorplanning*, simulação, *placement* automático e roteamento de interconexões. Esta ferramenta é usada para implementar e otimizar circuitos em FPGA. A entrada desta ferramenta é a especificação do circuito descrito em uma linguagem de alto nível do tipo VHDL. Em seguida

este circuito a ferramenta irá posicionar os CLB e IOB dentro do FPGA (fase de *(placement)*), para depois rotear de todas as conexões do circuito. Por fim o projeto os arquivos de configuração do circuito são transferidos para a memória interna do FPGA, e as interconexões programadas para a implementação.

Os dispositivos de FPGA de um fabricante são divididos em famílias, as mais tradicionais são: XC4000 e XC6200.

B.2 Família XC4000

A família XC4000 os CLB são ligados por canais de roteamento verticais e horizontais e são rodeados por IOB. São baseados em "look-up tables", que são arrays de memórias de 1 bit de largura. Nesta família cada CLB possui dois *flip-flops*, um com quatro entradas e outro de três entradas, como mostra a Figura B.2. O elevado número de flip-flops na família XC4000 é um convida ao desenvolvimento de projetos com paralelismo do tipo pipelines, usado para aumentar o desempenho do sistema.

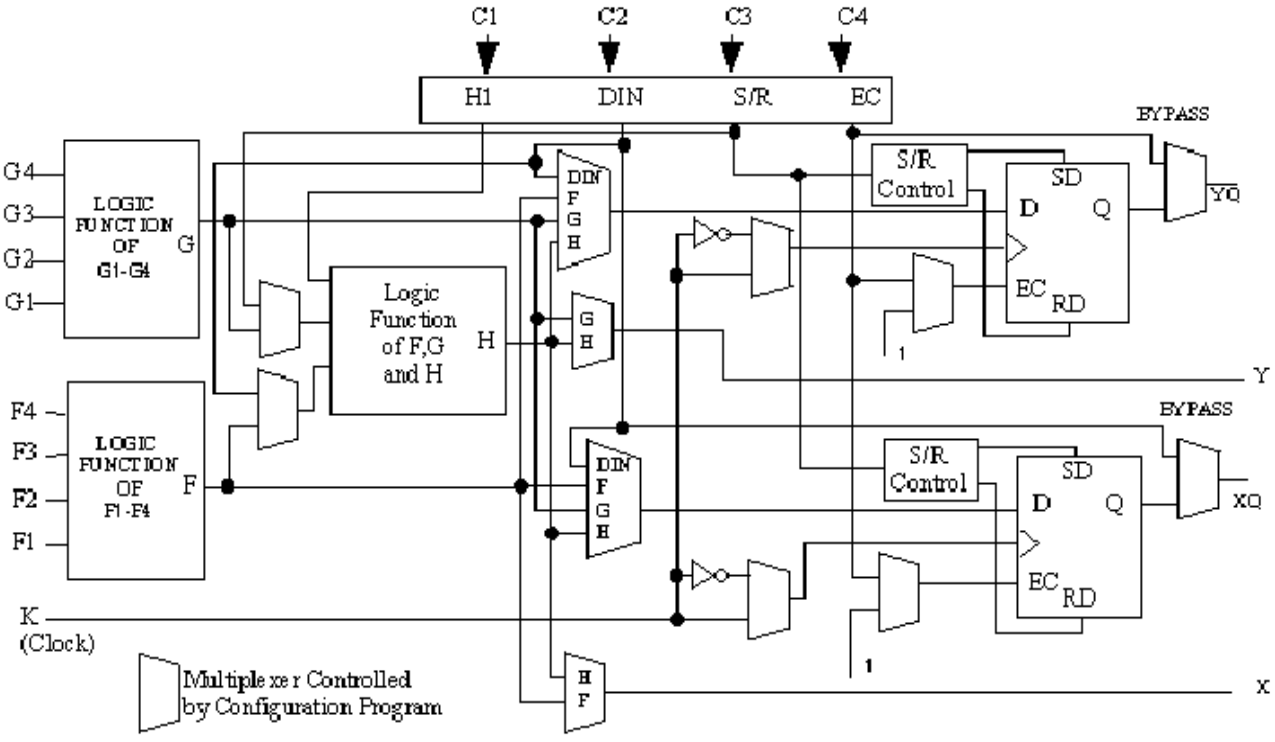


Figura B.2: Arquitetura do CLB da família XC4000

As unidades provedoras de sinal de um CLB, chamadas de *fast carry logic*, dispõem de uma aritmética lógica capaz de gerar rapidamente os sinais de *carry* e *borrow* em circuitos aritméticos, como mostra a Figura B.3.

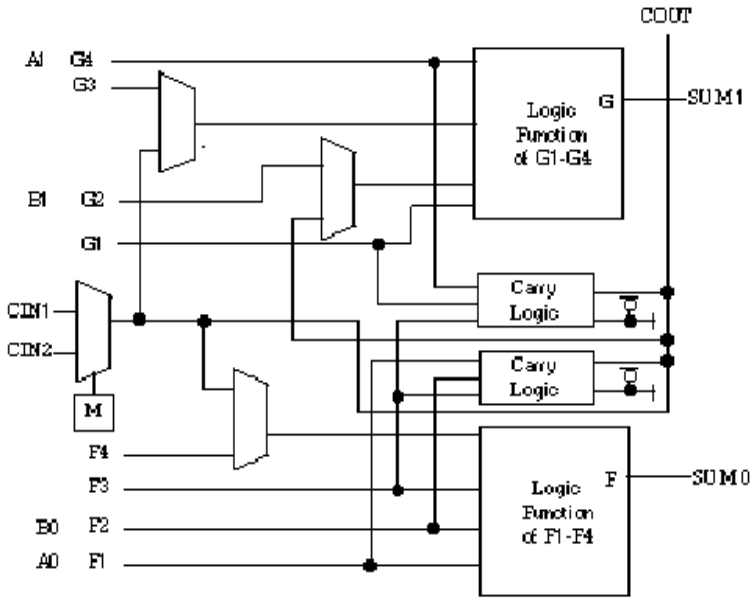


Figura B.3: Arquitetura do *fast carry logic* da família XC4000

A Figura B.4 apresenta a arquitetura do IOB da família XC4000 que serve de interface entre os pinos externos e a lógica interna. Cada IOB controla um pino e pode ser definido como uma entrada, uma saída ou um sinal bidirecional. As interconexões programáveis são compostas de segmentos de metais com pontos e matrizes de chaveamento para implementar o roteamento de CLB, de IOB (denominada de *VersaRing*) e global. O roteamento para o CLB é associado a cada linha e coluna do matriz do CLB.

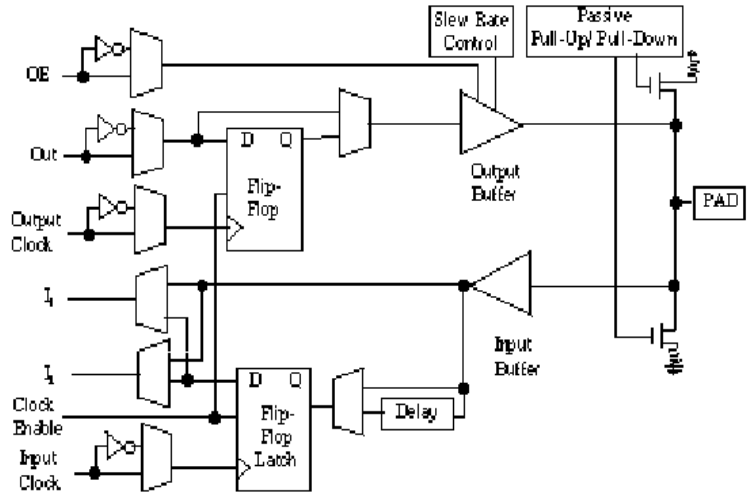


Figura B.4: Arquitetura do IOB da família XC4000

As interconexões são caracterizadas pelos tamanhos dos segmentos: *single-length lines* atravessam o CLB

usando matrizes de chaveamento (Figura B.5); *dual-length lines* atravessam dois CLB e também usam matrizes de chaveamento (Figura B.6); e *long lines* atravessam o FPGA inteiro no comprimento ou na largura e não utilizam matrizes de chaveamento (Figura B.7).

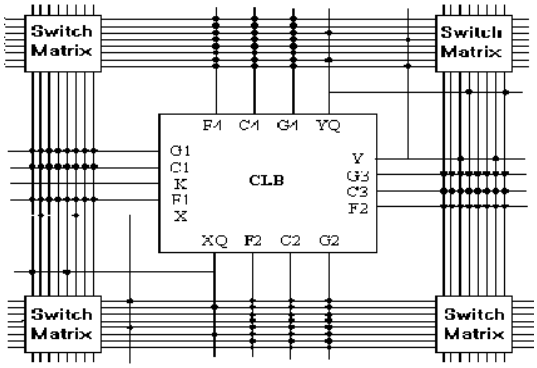


Figura B.5: Conexões *single-length line*

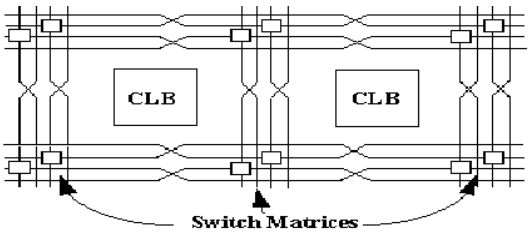


Figura B.6: Conexões *dual-length line*

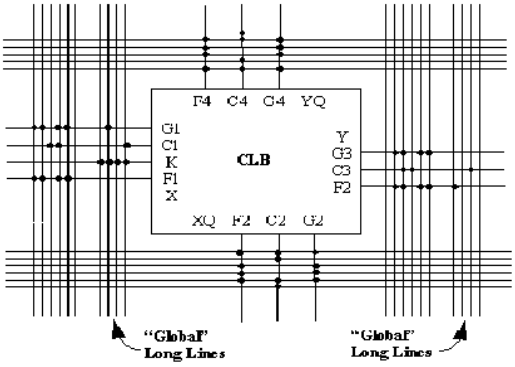


Figura B.7: Conexões *long-length line*

Embora reconfiguráveis, os componentes da família XC4000 de FPGA não são parcialmente reconfiguráveis, ou seja, sua reconfiguração implica numa reconfiguração total de todo seu circuito.

B.3 Família XC6200

A família XC6200 se diferencia da XC4000 por suportar reconfiguração parcial e total de seus CLB. A arquitetura XC6200 pode ser vista como uma hierarquia. No nível mais baixo da hierarquia há uma grande quantidade de CLB simples (Figura B.8), denominadas de *sea of gates*. Nesta família o CLB é individualmente programável para implementar um *flip-flop* do tipo D e uma função lógica tal como um multiplexador ou uma porta lógica. Qualquer CLB também pode ser configurado para executar uma função puramente combinacional.

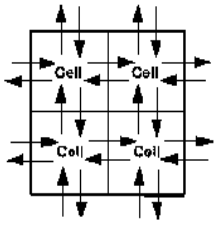


Figura B.8: Hierarquia da família XC6200

Os CLB vizinhos são agrupados em blocos de 4x4 que formam as células de uma matriz de CLB, estes CLB comunicando-se entre si, como mostra a Figura. Um agrupamento 4x4 das células da matriz 4x4 de CLB forma uma célula com 16x16 CLB, a qual é exibida na Figura B.9.

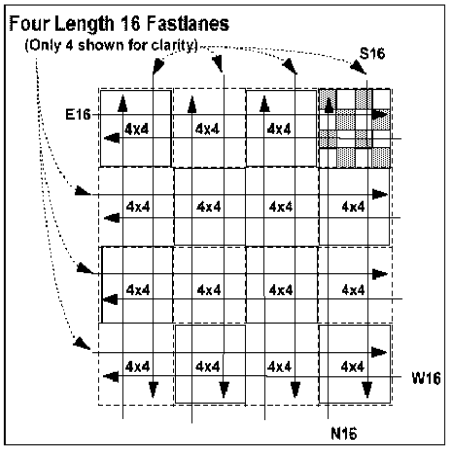


Figura B.9: Hierarquia da família XC6200

A arquitetura XC6200 permite que registros dentro de um projeto do usuário possam ser sincronizados por pulsos de *clocks* diferentes e serem limpos por pulsos *clears* assíncronos e diferentes. Os pulsos de *clock* e *clear* podem ser fornecidos por qualquer pino de entrada e saída, ou serem gerados internamente a partir da lógica do usuário.

Apêndice C

Provas do Capítulo 3

C.1 Corpo dos Racionais

Lema C.1.1. *(Associatividade da Operação de Adição dos Números Racionais)* -

$\forall x, y, z : \mathbb{Q} \bullet adicao\mathbb{Q}(x, adicao\mathbb{Q}(y, z)) = adicao\mathbb{Q}(adicao\mathbb{Q}(x, y), z)$

Prova:

1. Teorema
   $\forall x, y, z : \mathbb{Q} \bullet adicao\mathbb{Q}(x, adicao\mathbb{Q}(y, z)) = adicao\mathbb{Q}(adicao\mathbb{Q}(x, y), z)$

2. AdicaoRacionalDef [$x := y, y := z$];
   $\forall x, y, z : \mathbb{Q} \bullet adicao\mathbb{Q}(x, (first(y) * second(z) + first(z) * second(y), second(y) * second(z))) =$
   $adicao\mathbb{Q}(adicao\mathbb{Q}(x, y), z)$

3. AdicaoRacionalDef [$y := (first(y) * second(z) + first(z) * second(y), second(y) * second(z))$]
   $\forall x, y, z : \mathbb{Q} \bullet (first(x) * second(y) * second(z) + first(y) * second(x) * second(z) +$
   $first(z) * second(x) * second(y), second(x) * second(y) * second(z)) = adicao\mathbb{Q}(adicao\mathbb{Q}(x, y), z)$

4. AdicaoRacionalDef
   $\forall x, y, z : \mathbb{Q} \bullet (first(x) * second(y) * second(z) + first(y) * second(x) * second(z) +$
   $first(z) * second(x) * second(y), second(x) * second(y) * second(z)) =$
   $adicao\mathbb{Q}((first(x) * second(y) + first(y) * second(x), second(x) * second(y)), z)$

5. AdicaoRacionalDef
   $[x := (first(x) * second(y) + first(y) * second(x), second(x) * second(y)), y := z]$
   $True.\square$

Lema C.1.2. *(Elemento Neutro da Operação de Adição dos Números Racionais) -*

$\forall x : \mathbb{Q} \bullet adicao\mathbb{Q}(x,(0,1)) = adicao\mathbb{Q}((0,1),x) = x$

Prova:

1. Teorema
   $\forall x, y, z : \mathbb{Q} \bullet adicao\mathbb{Q}(x,(0,1)) = adicao\mathbb{Q}((0,1),x) = x$
2. AdicaoRacionalDef [y := (0,1)]
   $\forall x : \mathbb{Q} \bullet adicao\mathbb{Q}((0,1),x) = x$
3. AdicaoRacionalDef [x := (0,1) , y := x]
   $True.\square$

Lema C.1.3. *(Operação Inversa da Operação de Adição dos Números Racionais) -*

$\forall x : \mathbb{Q} \bullet \exists y : \mathbb{Q} \bullet (adicao\mathbb{Q}(x,y),(0,1)) \in equivalenteR$

Prova:

1. Teorema
   $\forall x : \mathbb{Q} \bullet \exists y : \mathbb{Q} \bullet (adicao\mathbb{Q}(x,y),(0,1)) \in mesmo\mathbb{Q}$
2. instantiate y == (-p,q)
   $\forall x : \mathbb{Q} \bullet (adicao\mathbb{Q}((p,q),(-p,q)),(0,1)) \in mesmo\mathbb{Q}$
3. AdicaoRacionalDef [y := (-p,q)]
   $\forall x : \mathbb{Q} \bullet ((p*q-p*q,q*q),(0,1)) \in mesmo\mathbb{Q}$
4. EquivalenciaRacionalDef [x := (0, q * q) , y := (0,1)]
   $True.\square$

Lema C.1.4. *(Operação de Adição dos Números Racionais é Univoca) -*

$\forall x, y : \mathbb{Q} \bullet adicao\mathbb{Q}(x,y) = adicao\mathbb{Q}(y,x)$

Prova:

1. Teorema
   $\forall x, y : \mathbb{Q} \bullet adicao\mathbb{Q}(x,y) = adicao\mathbb{Q}(y,x)$
2. AdicaoRacionalDef
   $\forall x, y : \mathbb{Q} \bullet (first(x) * second(y) + first(y) * second(x), second(x) * second(y)) = adicao\mathbb{Q}(y,x)$

3. AdicaoRacionalDef [x := y , y := x]
   $True.\square$

Lema C.1.5. *(Associatividade da Operação de Multiplicação dos Números Racionais) -*

$\forall x, y, z : \mathbb{Q} \bullet multiplicacao\mathbb{Q}(x, multiplicacao\mathbb{Q}(y, z)) = multiplicacao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), z)$

Prova:

1. Teorema
   $\forall x, y, z : \mathbb{Q} \bullet multiplicacao\mathbb{Q}(x, multiplicacao\mathbb{Q}(y, z)) = multiplicacao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), z)$
2. MultiplicacaoRacionalDef [x := y , y := z]
   $\forall x, y, z : \mathbb{Q} \bullet multiplicacao\mathbb{Q}(x, (first(y) * first(z), second(y) * second(z)) =$
   $multiplicacao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), z)$
3. MultiplicacaoRacionalDef [x := x , y := (first(y)*first(z),second(y)*second(z))]
   $\forall x, y, z : \mathbb{Q} \bullet (first(x) * first(y) * first(z), second(x) * second(y) * second(z)) =$
   $multiplicacao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), z)$
4. MultiplicacaoRacionalDef
   $\forall x, y, z : \mathbb{Q} \bullet (first(x) * first(y) * first(z), second(x) * second(y) * second(z)) =$
   $multiplicacao\mathbb{Q}((first(x) * first(y), second(x) * second(y)), z)$
5. MultiplicacaoRacionalDef [x :=(first(x)*first(y),second(x)*second(y)) , y := z]
   $True.\square$

Lema C.1.6. *(A Operação de Multiplicação dos Números Racionais é Distributiva em Relação a Operação de Adição dos Números Racionais) -*

$\forall x, y, z : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(x, adicao\mathbb{Q}(y, z)),$

$adicao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), multiplicacao\mathbb{Q}(x, z))) \in mesmo\mathbb{Q}$ *e*

$\forall x, y, z : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(adicao\mathbb{Q}(y, z), x),$

$adicao\mathbb{Q}(multiplicacao\mathbb{Q}(y, x), multiplicacao\mathbb{Q}(z, x))) \in mesmo\mathbb{Q}$*;*

Prova:

1. Teorema
   $\forall x, y, z : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(x, adicao\mathbb{Q}(y, z)),$
   $adicao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), multiplicacao\mathbb{Q}(x, z))) \in mesmo\mathbb{Q}$

2. AdicaoRacionalDef [x := y , y := z]

   $\forall x, y, z : \mathbb{Q} \bullet$

   $(multiplicacao\mathbb{Q}(x, (first(y) * second(z) + first(z) * second(y), second(y) * second(z))),$

   $adicao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), multiplicacao\mathbb{Q}(x, z))) \in mesmo\mathbb{Q}$

3. MultiplicacaoRacionalDef [y:=(first(y)*second(z)+first(z)*second(y),second(y)*second(z))]

   $\forall x, y, z : \mathbb{Q} \bullet$

   $((first(x) * first(y) * second(z) + first(z) * second(y), second(x) * second(y) * second(z)),$

   $adicao\mathbb{Q}(multiplicacao\mathbb{Q}(x, y), multiplicacao\mathbb{Q}(x, z))) \in mesmo\mathbb{Q}$

4. MultiplicacaoRacionalDef

   $\forall x, y, z : \mathbb{Q} \bullet$

   $((first(x) * first(y) * second(z) + first(z) * second(y), second(x) * second(y) * second(z)),$

   $adicao\mathbb{Q}((first(x) * first(y), second(x) * second(y)), multiplicacao\mathbb{Q}(x, z))) \in mesmo\mathbb{Q}$

5. use MultiplicacaoRacionalDef [y := z]

   $\forall x, y, z : \mathbb{Q} \bullet$

   $((first(x) * first(y) * second(z) + first(z) * second(y), second(x) * second(y) * second(z)),$

   $adicao\mathbb{Q}((first(x) * first(y), second(x) * second(y)),$

   $(first(x) * first(y), second(x) * second(y)))) \in mesmo\mathbb{Q}$

6. AdicaoRacionalDef [x := (first(x)*first(y),second(x)*second(y)) ,

   y := (first(x)*first(z),second(x)*second(z))]

   $\forall x, y, z : \mathbb{Q} \bullet$

   $((first(x) * first(y) * second(z) + first(z) * second(y), second(x) * second(y) * second(z)),$

   $((first(x) * first(y) * second(x) * second(y) + first(x) * first(y) * second(x) * second(y),$

   $second(x) * second(y) * second(x) * second(y))) \in mesmo\mathbb{Q}$

7. MesmoRacionalDef [x := (first(x)*first(y)*second(z)+first(z)*second(y),

   second(x)*second(y)*second(z)), y := (first(x)*first(y)*second(x)*second(y)+

   first(x)*first(y)*second(x)*second(y),second(x)*second(y)*second(x)*second(y))]

   $True$

e

1. Teorema

   $\forall x, y, z : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(adicao\mathbb{Q}(y, z), x),$

   $adicao\mathbb{Q}(multiplicacao\mathbb{Q}(y, x), multiplicacao\mathbb{Q}(z, x))) \in mesmo\mathbb{Q}$

2. AdicaoRacionalDef [x := y , y := z]
   $\forall x, y, z : \mathbb{Q} \bullet$
   $(multiplicacao\mathbb{Q}((first(y) * second(z) + first(z) * second(y), second(y) * second(z)), x),$
   $adicao\mathbb{Q}(multiplicacao\mathbb{Q}(y, x), multiplicacao\mathbb{Q}(z, x))) \in mesmo\mathbb{Q}$

3. MultiplicacaoRacionalDef [x := (first(y)*second(z)+first(z)*second(y),
   second(y)*second(z)) , y := x]; $\forall x, y, z : \mathbb{Q} \bullet$
   $((first(y)*second(z)*first(x)+first(z)*second(y)*first(x), second(y)*second(z)*second(x)),$
   $adicao\mathbb{Q}(multiplicacao\mathbb{Q}(y, x), multiplicacao\mathbb{Q}(z, x))) \in mesmo\mathbb{Q}$

4. MultiplicacaoRacionalDef [x := y , y := x]; $\forall x, y, z : \mathbb{Q} \bullet$
   $((first(y)*second(z)*first(x)+first(z)*second(y)*first(x), second(y)*second(z)*second(x)),$
   $adicao\mathbb{Q}((first(y) * first(x), second(y) * second(x)), multiplicacao\mathbb{Q}(z, x))) \in mesmo\mathbb{Q}$

5. MultiplicacaoRacionalDef [x := z , y := x]; $\forall x, y, z : \mathbb{Q} \bullet$
   $((first(y)*second(z)*first(x)+first(z)*second(y)*first(x), second(y)*second(z)*second(x)),$
   $adicao\mathbb{Q}((first(y)*first(x), second(y)*second(x)), (first(z)*first(x), second(z)*second(x)))) \in$
   $mesmo\mathbb{Q}$

6. AdicaoRacionalDef [x := (first(y)*first(x),second(y)*second(x)),
   y := (first(z)*first(x),second(z)*second(x))]
   $\forall x, y, z : \mathbb{Q} \bullet$
   $((first(y)*second(z)*first(x)+first(z)*second(y)*first(x), second(y)*second(z)*second(x)),$
   $(first(y) * first(x) * second(z) * second(x) + first(z) * first(x) * second(y) * second(x),$
   $second(y) * second(x) * second(z) * second(x))) \in mesmo\mathbb{Q}$

7. MesmoRacionalDef [(first(y)*second(z)*first(x)+first(z)*second(y)*first(x),
   second(y)*second(z)*second(x)), y := (first(y)*first(x)*second(z)*second(x)+
   first(z)*first(x)*second(y)*second(x), second(y)*second(x)*second(z)*second(x))];
   $True.\square$

Lema C.1.7. *(Elemento Neutro da Operação de Multiplicação dos Números Racionais)* -
$\forall x : \mathbb{Q} \bullet multiplicacao\mathbb{Q}(x, (1, 1)) = multiplicacao\mathbb{Q}((1, 1), x) = x$

Prova:

1. Teorema
   $\forall x : \mathbb{Q} \bullet multiplicacao\mathbb{Q}(x, (1, 1)) = multiplicacao\mathbb{Q}((1, 1), x) = x$

2. MultiplicacaoRacionalDef [x := x , y := (1,1)]
   $\forall x : \mathbb{Q} \bullet (first(x), second(x)) = multiplicacao\mathbb{Q}((1,1), x) = x$

3. MultiplicacaoRacionalDef [x := (1,1) , y := x]
   $True.\square$

Lema C.1.8. *(A Operação de Multiplicação dos Números Racionais tem Inversa para os Números Diferentes de Zero)* - $\forall x : \mathbb{Q} \mid first(x) \neq 0 \bullet \exists y : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(x, y), (1,1)) \in mesmo\mathbb{Q}$

Prova:

1. Teorema
   $\forall x : \mathbb{Q} \mid first(x) \neq 0 \bullet \exists y : \mathbb{Q} \bullet (multiplicacao\mathbb{Q}(x, y), (1,1)) \in mesmo\mathbb{Q}$

2. instantiate y == (q,p)
   $\forall x : \mathbb{Q} \mid first(x) \neq 0 \bullet (multiplicacao\mathbb{Q}(x, (second(x), first(x))), (1,1)) \in mesmo\mathbb{Q}$

3. MultiplicacaoRacionalDef [y := (q,p)]
   $\forall x : \mathbb{Q} \mid first(x) \neq 0 \bullet ((first(x) * second(x), second(x) * first(x)), (1,1)) \in mesmo\mathbb{Q}$

4. MesmoRacionalDef [x := (first(x)*second(x),second(x)*first(x)) , y := (1,1)]
   $True.\square$

C.2 Grau de Adaptação do Gene

Lema C.2.1. *(Propriedade Requerida sobre o Gene Inócuo)* -

$\forall g : G \mid g \neq g_\lambda \bullet grau(g), grau(g_\lambda) \in maior\mathbb{Q}$

Prova: Dizer que $\forall g : G \mid g \neq g_\lambda \bullet grau(g), grau(g_\lambda) \in maior\mathbb{Q}$ é a mesma coisa que dizer que $\forall g : G \mid \mathsf{ran}\ g = E \bullet grau(g), grau(g_\lambda) \in maior\mathbb{Q}$, onde E são todos os subconjuntos do conjunto $\{A, B, C, D\}$ com cardinalidade dois.

Assim o teorema $\forall g : G \mid \mathsf{ran}\ g = \{A, B\} \bullet (grau(g), grau(g_\lambda)) \in maior\mathbb{Q}$ é verdade porque:

1. GrauInocuoDef
   $\forall g : G \mid \mathsf{ran}\ g = \{A, B\} \bullet (grau(g), (0,1)) \in maior\mathbb{Q}$

2. GrauABDef
   $\forall g : G \mid \mathsf{ran}\ g = \{A, B\} \bullet ((1,3), (0,1)) \in maior\mathbb{Q}$

3. MaiorRacionalDef [x := (1,3) , y := (0,1)]
   $True$

De forma similar prova-se que o teorema $\forall g : G \mid \mathsf{ran}\ g = E \bullet grau(g), grau(g_\lambda) \in maior\mathbb{Q}$, é verdadeiro para os outros subconjuntos do conjunto $\{A, B, C, D\}$ de cardinalidade dois.□

C.3 Mesma Característica

Lema C.3.1. *(A Relação Mesma Característica é Reflexiva)* - $\forall g : G \bullet (g, g) \in mesma$

Prova:

1. $Teorema$
   $\forall g : G \bullet (g, g) \in mesma$

2. $mesmaDef[g_1 := g, g_2 := g]$
   $\forall g : G \mid (g, g) \in mesma \bullet \exists G_1 : atributoRelevante \bullet g \in G_1$

3. $atributoRelevanteDef[g_1 := g, g_2 := g]$
   $True$.□

Lema C.3.2. *(A Relação Mesma Característica é Comutativa)* -

$\forall g_1, g_2 : G \mid (g_1, g_2) \in mesma \bullet (g_2, g_1) \in mesma$

Prova:

1. $Teorema$
   $\forall g_1, g_2 : G \mid (g_1, g_2) \in mesma \bullet (g_2, g_1) \in mesma$

2. $mesmaDef$
   $\forall g_1, g_2 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \bullet$
   $(g_2, g_1) \in mesma$

3. $mesmaDef[g_1 := g_2, g_2 := g_1]$;
   $\forall g_1, g_2 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \bullet \exists G_2 : atributoRelevante \bullet g_1 \in G_2 \land g_2 \in G_2$

4. $instantiate\ G_2 == G_1$
   $True$.□

Lema C.3.3. *(A Relação Mesma Característica é Transitiva) -*

$\forall g_1, g_2, g_3 : G \mid (g_1, g_2) \in mesma \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$

Prova:

1. $Teorema$
   $\forall g_1, g_2, g_3 : G \mid (g_1, g_2) \in mesma \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$

2. $mesmaDef$
   $\forall g_1, g_2, g_3 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$

3. $atributoRelevanteDef$
   $\forall g_1, g_2, g_3 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2) \Leftrightarrow$
   $(\exists G_2 : atributoRelevante \bullet g_1 \in G_2 \land g_2 \in G_2)) \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$

4. $instantiate\, G_2 == G_1;$
   $\forall g_1, g_2, g_3 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \land$
   $(raiz(g_1) = raiz(g_2)) \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$

5. $mesmaDef[g_1 := g_2, g_2 := g_3]$
   $\forall g_1, g_2, g_3 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevante \land g_2 \in G_2 \land g_3 \in G_2 \bullet (g_1, g_3) \in mesma$

6. $atributoRelevanteDef[g_1 := g_2, g_2 := g_3]$
   $\forall g_1, g_2, g_3 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevante \land g_2 \in G_2 \land g_3 \in G_2 \land$
   $(raiz(g_2) = raiz(g_3) \Leftrightarrow (\exists G_3 : atributoRelevante \bullet g_2 \in G_3 \land g_3 \in G_3)) \bullet (g_1, g_3) \in mesma$

7. $instantiate\, G_3 == G_2;$
   $\forall g_1, g_2, g_3 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevante \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \bullet (g_1, g_3) \in mesma$

8. $mesmaDef[g_2 := g_3]$
   $\forall g_1, g_2, g_3 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevante \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \bullet$
   $(\exists G_4 : atributoRelevante \bullet g_1 \in G_4 \land g_3 \in G_4)$

9. $atributoRelevanteDef[g_2 := g_3]$
   $\forall g_1, g_2, g_3 : G \mid G_1 \in atributoRelevante \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$

$G_2 \in atributoRelevante \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \land$
$G_3 \in atributoRelevante \land g_1 \in G_3 \land g_3 \in G_3 \land (raiz(g_1) = raiz(g_3)) \bullet$
$(\exists G_4 : atributoRelevante \bullet g_1 \in G_4 \land g_3 \in G_4)$

10. $instantiate\, G_4 == G_3$
    $True.\square$

C.4 Gene Dominante

Lema C.4.1. *(A Função Gene Dominanate é Reflexiva)* - $\forall g : G \bullet domi(g, g) = g$

Prova:

1. Teorema
   $\forall g : G \bullet domi(g, g) = g$
2. domiIgualDef [$g_1 := g, g_2 := g$]
   $\forall g : G \mid (g, g) \in mesma \land (grau(g), grau(g)) \in igualR \bullet domi(g, g) = g$
3. mesmaReflexiva
   $\forall g : G \mid (grau(g), grau(g)) \in igualR \bullet domi(g, g) = g$
4. $\forall g : G \bullet \exists r : \mathbb{Q} \bullet grau(g) = r$
5. MesmoRacionalDef [x := r , y := r]
   $True.\square$

Lema C.4.2. *(A Função Gene Dominante é Associativa)* -

$\forall g_1, g_2, g_3 : G \bullet domi(g_1, domi(g_2, g_3)) = domi(domi(g_1, g_2), g_3)$

Prova: A prova de que $\forall g_1, g_2, g_3 : G \bullet domi(g_1, domi(g_2, g_3)) = domi(domi(g_1, g_2), g_3)$ requer a análise de todas as possíveis relações entre o grau de adaptação dos genes envolvidos. Para o caso em que o grau de adaptação entre os genes envolvidos é o mesmo, tem-se que:

1. Teorema
   $\forall g_1, g_2, g_3 : G \mid (grau(g_1), grau(g_2)) \in mesmo\mathbb{Q} \land$
   $(grau(g_1), grau(g_3)) \in mesmo\mathbb{Q} \land (grau(g_2), grau(g_3)) \in mesmo\mathbb{Q} \bullet$
   $domi(g_1, domi(g_2, g_3)) = domi(domi(g_1, g_2), g_3)$

2. domiIgualDef

   $\forall g_1, g_2, g_3 : G \mid (grau(g_1), grau(g_2)) \in mesmo\mathbb{Q} \wedge$
   $(grau(g_1), grau(g_3)) \in mesmo\mathbb{Q} \wedge (grau(g_2), grau(g_3)) \in mesmo\mathbb{Q} \wedge$
   $((g_1, g_2) \in mesma \implies domi(g_1, g_2) = g_1) \bullet domi(g_1, domi(g_2, g_3)) = domi(domi(g_1, g_2), g_3)$

3. mesmaDef

   $\forall g_1, g_2, g_3 : G \mid (grau(g_1), grau(g_2)) \in mesmo\mathbb{Q} \wedge$
   $(grau(g_1), grau(g_3)) \in mesmo\mathbb{Q} \wedge (grau(g_2), grau(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(g_1, g_2) \in mesma \wedge domi(g_1, g_2) = g_1 \bullet domi(g_1, domi(g_2, g_3)) = domi(g_1, g_3)$

4. domiIgualDef [$g_2 := g_3$]

   $\forall g_1, g_2, g_3 : G \mid (grau(g_1), grau(g_2)) \in mesmo\mathbb{Q} \wedge$
   $(grau(g_1), grau(g_3)) \in mesmo\mathbb{Q} \wedge (grau(g_2), grau(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(g_1, g_2) \in mesma \wedge domi(g_1, g_2) = g_1 \wedge ((g_1, g_3) \in mesma \implies domi(g_1, g_3) = g_1) \bullet$
   $domi(g_1, domi(g_2, g_3)) = domi(g_1, g_3)$

5. mesmaDef [$g_2 := g_3$]

   $\forall g_1, g_2, g_3 : G \mid (grau(g_1), grau(g_2)) \in mesmo\mathbb{Q} \wedge$
   $(grau(g_1), grau(g_3)) \in mesmo\mathbb{Q} \wedge (grau(g_2), grau(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(g_1, g_2) \in mesma \wedge domi(g_1, g_2) = g_1 \wedge (g_1, g_3) \in mesma \wedge domi(g_1, g_3) = g_1) \bullet$
   $domi(g_1, domi(g_2, g_3)) = g_1$

6. domiIgualDef [$g_1 := g_2$, $g_2 := g_3$]

   $\forall g_1, g_2, g_3 : G \mid (grau(g_1), grau(g_2)) \in mesmo\mathbb{Q} \wedge$
   $(grau(g_1), grau(g_3)) \in mesmo\mathbb{Q} \wedge (grau(g_2), grau(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(g_1, g_2) \in mesma \wedge domi(g_1, g_2) = g_1 \wedge (g_1, g_3) \in mesma \wedge domi(g_1, g_3) = g_1) \wedge$
   $((g_2, g_3) \in mesma \implies domi(g_2, g_3) = g_2) \bullet domi(g_1, domi(g_2, g_3)) = g_1$

7. mesmaDef [$g_1 := g_2$, $g_2 := g_3$]

   $True$

Seguindo este mesmo raciocíneo obtém-se que a propriedade associativa da função $domi$ é verdadeira qualquer que seja a relação existente entre o grau de adapção dos genes analisados..□

C.5 Fecundação

Lema C.5.1. *(A Função Fecunação é Reflexiva)* - $\forall c : CROMOSSOMO \bullet (fec(c, c), c) \in \equiv_C$

Prova: O enfraquecimento da propriedade reflexiva da função de fecundação deve-se ao fato de que quando dois genes quaisquer pertencentes ao cromossomo c não puderem ser comparados, então o gene retornado pela função $domi$ é g_λ, cuja a presença não interfere na identidade do cromossomo c. A prova da propriedade reflexiva da função de fecundação será realizada em três passos cada um representando uma das possibilidades de retorno da função $domi$, que poderá ser g_λ, g_1 ou g_2. A prova de que a função fec é reflexiva quando os genes submetidos a função $domi$ não pertencem a uma mesma caracetrístia é exibida a seguir.

1. Teorema
   $\forall c : CROMOSSOMO \bullet \forall g1, g2 : G \mid g1 \in c \land g2 \in c \land domi(g1, g2) = g_\lambda \bullet g_\lambda \in fec(c, c)$

2. FecDef [c1 := c , c2 := c]
   $True$

Alterando o gene retornado pela função $domi$ no Lema acima e seguindo o mesmo passo prova-se que a função fec é reflexiva.□

C.6 Troca

Lema C.6.1. *(A Função Troca pode ser Definida como a Composição das Funções Inserção e Supressão)* - $\forall c : CROMOSSOMO;\ G_1, G_2 : \mathbb{P}(G) \bullet troc(c, G_1, G_2) = del(ins(c, G_1), G_2)$

Prova: A prova requer a análise das alternativas: $trocSimSimDef$, $trocSimNaoDef$, $trocNaoSimDef$ e $trocNaoNaoDef$; utilizadas na definição da função de troca. Para a alternativa $trocSimSimDef$, obtém-se que:

1. Teorema
   $\forall c : CROMOSSOMO;\ G_1, G_2 : \mathbb{P}(G) \mid c \cup G_1 \in afc \land (c \cup G_1) \setminus G_2 \bullet$
   $troc(c, G_1, G_2) = del(ins(c, G_1), G_2)$

2. trocSimSimDef [c := c]
   $\forall c : CROMOSSOMO;\ G_1, G_2 : \mathbb{P}(G) \mid c \cup G_1 \in afc \land (c \cup G_1) \setminus G_2 \land$
   $troc(c, G_1, G_2) = (c \cup G_1) \setminus G_2 \bullet (c \cup G_1) \setminus G_2 = del(ins(c, G_1), G_2)$

3. insSimDef [c := c , G := G_1]
   $\forall c : CROMOSSOMO;\ G_1, G_2 : \mathbb{P}(G) \mid c \cup G_1 \in afc \land (c \cup G_1) \setminus G_2 \land$
   $troc(c, G_1, G_2) = (c \cup G_1) \setminus G_2 \land ins(c, G_1) = c \cup G_1 \bullet (c \cup G_1) \setminus G_2 = del(c \cup G_1, G_2)$

4. insSimDef delSimDef [c := $c \cup G_1$, G := G_2]
   $True$

A prova segue trivialmente por analogia das alternativas da função troca com as correspondentes alternativas das função de inserção e supressão.□

Apêndice D

Provas do Capítulo 5

D.1 Problema de construção de árvores filogenéticas

D.1.1 Homologia

Lema D.1.1. $\forall b_1, b_2 : BI \bullet (homologia(b_1, b_2), (0, 1)) \in maior\mathbb{Q}$.

Prova: A prova requer a análise das alternativas usadas na definição da função $homologia$. Para a alternativa em que uma das bases envolvidas é e_λ, tem-se que:

1. $Teorema$
   $\forall b_1, b_2 : BI \mid b_1 \neq b_2 \wedge b_2 = e_\lambda \bullet (homologia(b_1, b_2), (0, 1)) \in maior\mathbb{Q}$
2. $HomologiaInocuoDef$
   $\forall b_1, b_2 : BI \mid b_1 \neq b_2 \wedge b_2 = e_\lambda \bullet ((4, 1), (0, 1)) \in maior\mathbb{Q}$
3. $MaiorRacionalDef[x := (4, 1), y := (0, 1)]$
   $True$

De modo análogo mostra-se que o teorema $\forall b_1, b_2 : BI \bullet (homologia(b_1, b_2), (0, 1)) \in maior\mathbb{Q}$ é verdadeiro para as outras alternativas apresentadas na definição da função $homologia$. □

D.1.2 Grau de Adaptação

Lema D.1.2. $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), grauI(gI_\lambda)) \in maior\mathbb{Q}$.

Prova: Esta prova usa dois teoremas: um para estabelecer que a homologia entre quaisquer dois elementos da base é sempre maior do que zero (Lema D.1.1) e, outro que o resultado da adição racional de dois números racionais maior do que zero é um número racional também maior do que zero (prova no Apêndice A).

1. $Teorema$
   $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), grauI(gI_\lambda)) \in maior\mathbb{Q}$

2. $GrauIInocuoDef$
   $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), (0,1)) \in maior\mathbb{Q}$

3. $GrauINaoInocuoDef$
   $\forall g : GI \mid g \neq gI_\lambda \bullet (adicao\mathbb{Q}(homologia(direita(g), raiz(g)),$
   $homologia(esquerda(g), raiz(g))), (0,1)) \in maior\mathbb{Q}$

4. $RaizDef$
   $\forall g : GI \mid g \neq gI_\lambda \bullet$
   $(adicao\mathbb{Q}(homologia(direita(g), head(tail(g))),$
   $homologia(esquerda(g), head(tail(g)))), (0,1)) \in maior\mathbb{Q}$

5. $DireitaDef$
   $\forall g : GI \mid g \neq gI_\lambda \bullet (adicao\mathbb{Q}(homologia(last(g), head(tail(g))),$
   $homologia(esquerda(g), head(tail(g)))), (0,1)) \in maior\mathbb{Q}$

6. $EsquerdaDef$
   $\forall g : GI \mid g \neq gI_\lambda \bullet (adicao\mathbb{Q}(homologia(last(g), head(tail(g))),$
   $homologia(head(g), head(tail(g)))), (0,1)) \in maior\mathbb{Q}$

7. $HomologiasMaiorZero[b_1 := (last(g), b_2 := head(tail(g))]$
   $\forall g : GI \mid g \neq gI_\lambda \wedge (homologia(last(g), head(tail(g))), (0,1)) \in maior\mathbb{Q} \bullet$
   $(adicao\mathbb{Q}(homologia(last(g), head(tail(g))),$
   $homologia(head(g), head(tail(g)))), (0,1)) \in maior\mathbb{Q}$

8. $HomologiasMaiorZero[b_1 := head(g), b_2 := head(tail(g))]$
   $\forall g : GI \mid g \neq gI_\lambda \wedge (homologia(last(g), head(tail(g))), (0,1)) \in maior\mathbb{Q}$
   $\wedge (homologia(head(g), head(tail(g))), (0,1)) \in maior\mathbb{Q} \bullet$
   $(adicao\mathbb{Q}(homologia(last(g), head(tail(g))),$
   $homologia(head(g), head(tail(g)))), (0,1)) \in maior\mathbb{Q}$

9. $AdicaoMaiorZero[x := homologia(last(g), head(tail(g))),$
   $y := homologia(head(g), head(tail(g)))]$
   $True.\square$

D.1.3 Mesma Característica

Lema D.1.3. $\forall g : GI \bullet (g, g) \in mesmaI$

Prova:

1. $Teorema$
   $\forall g : GI \bullet (g, g) \in mesmaI$
2. $mesmaIDef[g_1 := g, g_2 := g]$
   $\forall g : GI \mid (g, g) \in mesmaI \bullet \exists G_1 : atributoRelevanteI \bullet g \in G_1$
3. $atributoRelevanteIDef[g_1 := g, g_2 := g]$
   $True.\square$

Lema D.1.4. $\forall g_1, g_2 : GI \mid (g_1, g_2) \in mesmaI \bullet (g_2, g_1) \in mesmaI$

Prova:

1. $Teorema$
   $\forall g_1, g_2 : GI \mid (g_1, g_2) \in mesmaI \bullet (g_2, g_1) \in mesmaI$
2. $mesmaIDef$
   $\forall g_1, g_2 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \bullet (g_2, g_1) \in mesmaI$
3. $mesmaIDef[g_1 := g_2, g_2 := g_1]$;
   $\forall g_1, g_2 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \bullet$
   $\exists G_2 : atributoRelevanteI \bullet g_1 \in G_2 \land g_2 \in G_2$
4. $instantiate\ G_2 == G_1$
   $True.\square$

Lema D.1.5. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$

Prova:

1. $Teorema$
   $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$
2. $mesmaIDef$
   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (g_2, g_3) \in mesmaI \bullet$
   $(g_1, g_3) \in mesmaI$

3. $atributoRelevanteIDef$

   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2) \Leftrightarrow$
   $(\exists G_2 : atributoRelevanteI \bullet g_1 \in G_2 \land g_2 \in G_2)) \land (g_2, g_3) \in mesmaI \bullet$
   $(g_1, g_3) \in mesmaI$

4. $instantiate\, G_2 == G_1;$

   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land$
   $(raiz(g_1) = raiz(g_2)) \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$

5. $mesmaIDef[g_1 := g_2, g_2 := g_3]$

   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \bullet (g_1, g_3) \in mesmaI$

6. $atributoRelevanteIDef[g_1 := g_2, g_2 := g_3]$

   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \land$
   $(raiz(g_2) = raiz(g_3) \Leftrightarrow (\exists G_3 : atributoRelevanteI \bullet g_2 \in G_3 \land g_3 \in G_3)) \bullet (g_1, g_3) \in mesmaI$

7. $instantiate\, G_3 == G_2;$

   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \bullet (g_1, g_3) \in mesmaI$

8. $mesmaIDef[g_2 := g_3]$

   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \bullet$
   $(\exists G_4 : atributoRelevanteI \bullet g_1 \in G_4 \land g_3 \in G_4)$

9. $atributoRelevanteIDef[g_2 := g_3]$

   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \land$
   $G_3 \in atributoRelevanteI \land g_1 \in G_3 \land g_3 \in G_3 \land (raiz(g_1) = raiz(g_3)) \bullet$
   $(\exists G_4 : atributoRelevanteI \bullet g_1 \in G_4 \land g_3 \in G_4)$

10. $instantiate G_4 == G_3$

    $True.\square$

D.1.4 Gene Dominante

Lema D.1.6. $\forall g : GI \mid g = gI_\lambda \bullet domiI(g,g) = g \Leftrightarrow (g,g) \in mesmaI$

Prova: Esta prova será dividida em duas partes, na primeira mostra-se que a reflexividade da função $domiI$ é verdade quando $g = gI_\lambda$ e a segunda para o caso contrário.

1. $Teorema$
   $\forall g : GI \mid g = gI_\lambda \bullet domiI(g,g) = g \Leftrightarrow (g,g) \in mesmaI$

2. $mesmaIReflexiva[g := gI_\lambda]$
   $(gI_\lambda, gI_\lambda) \in mesmaI \bullet domiI(gI_\lambda, gI_\lambda) = gI_\lambda$

3. $DomiIIgualDef[g_1 := gI_\lambda, g_2 := gI_\lambda]$
   $(gI_\lambda, gI_\lambda) \in mesmaI \wedge (domiI(gI_\lambda, gI_\lambda) = gI_\lambda \Leftrightarrow (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}) \bullet$
   $(grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}$

4. $MesmoRacionalDef[x := grauI(gI_\lambda), y := grauI(gI_\lambda)]$
   $(gI_\lambda, gI_\lambda) \in mesmaI \wedge (domiI(gI_\lambda, gI_\lambda) = gI_\lambda \Leftrightarrow (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}) \wedge$
   $((\exists p, q : \mathbb{Z} \mid q > 0 \bullet grauI(gI_\lambda) = (p,q))) \bullet (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}$

5. $GrauIInocuoDef$
   $(gI_\lambda, gI_\lambda) \in mesmaI \wedge (domiI(gI_\lambda, gI_\lambda) = gI_\lambda \Leftrightarrow (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}) \wedge$
   $((\exists p, q : \mathbb{Z} \mid q > 0 \bullet grauI(gI_\lambda) = (p,q))) \wedge grauI(gI_\lambda) = (0,1) \bullet$
   $(grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}$

6. $instantiate\ p == 0$
   $(gI_\lambda, gI_\lambda) \in mesmaI \wedge (domiI(gI_\lambda, gI_\lambda) = gI_\lambda \Leftrightarrow (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}) \wedge$
   $((\exists q : \mathbb{Z} \mid q > 0 \bullet grauI(gI_\lambda) = (0,q))) \wedge grauI(gI_\lambda) = (0,1) \bullet$
   $(grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}$

7. $instantiate\ q == 1$
   $True$

A prova de que a função $domiI$ preserva a propriedade reflexiva para o caso de $g \neq gI_\lambda$ é análoga a prova acima, desdes que se utilise o teorema $GrauINaoInocuoDef$ no quinto passo desta prova com a correspondente instanciação das variáveis p e q. □

Lema D.1.7. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \wedge (g_1, g_3) \in mesmaI \wedge (g_2, g_3) \in mesmaI \bullet$
$domiI(g_1, domiI(g_2, g_3)) = domiI(domiI(g_1, g_2), g_3)$

Prova: A prova desta propriedade requer a avaliação das relações existentes entre o grau de adaptação dos genes envolvidos, que pode ser:

- $(grauI(g_1), grauI(g_2)) \in mesmo\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q}$;
- $(grauI(g_1), grauI(g_2)) \in menor\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q}$;
- $(grauI(g_1), grauI(g_2)) \in maior\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q}$;
- $(grauI(g_1), grauI(g_2)) \in menor\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in menor\mathbb{Q};\ e$
- $(grauI(g_1), grauI(g_2)) \in maior\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in maior\mathbb{Q}$.

No caso em que o grau de adptação de todos genes envolvidos é o mesmo temos:

1. $Teorema$
   $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \wedge (g_1, g_3) \in mesmaI \wedge (g_2, g_3) \in mesmaI \wedge$
   $(grauI(g_1), grauI(g_2)) \in mesmo\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(grauI(g_1), grauI(g_3)) \in mesmo\mathbb{Q} \bullet domiI(g_1, domiI(g_2, g_3)) = domiI(domiI(g_1, g_2), g_3)$
2. $DomiIIgualDef[g_1 := g_2, g_2 := g_3]$
   $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \wedge (g_1, g_3) \in mesmaI \wedge (g_2, g_3) \in mesmaI \wedge$
   $(grauI(g_1), grauI(g_2)) \in mesmo\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(grauI(g_1), grauI(g_3)) \in mesmo\mathbb{Q} \wedge domiI(g_2, g_3) = g_2 \bullet domiI(g_1, g_2) = domiI(domiI(g_1, g_2), g_3)$
3. $DomiIIgualDef$
   $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \wedge (g_1, g_3) \in mesmaI \wedge (g_2, g_3) \in mesmaI \wedge$
   $(grauI(g_1), grauI(g_2)) \in mesmo\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(grauI(g_1), grauI(g_3)) \in mesmo\mathbb{Q} \wedge domiI(g_2, g_3) = g_2 \wedge domiI(g_1, g_2) = g_1 \bullet g_1 = domiI(g_1, g_3)$
4. $DomiIIgualDef[g_2 := g_3]$
   $True$

Seguindo-se o raciocíneo acima exposto prova-se que a função $domiI$ é associativa. □

D.1.5 Fecundação

Lema D.1.8. $\forall c : CI \bullet (fecI(c, c), c) \in\equiv I_C$

Prova: A prova é identica a apresentada para o **Lema A.4.12**, desde que resguarada a instanciação dos dados e das funções envolvidas neste lema aos dados e funções apresentadas no Capítulo 5.□

D.2 Problema do controle de semáforos

D.2.1 Grau de Adaptação

Lema D.2.1. $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), grauI(gI_\lambda)) \in maior\mathbb{Q}$.

Prova: Esta prova usa dois teoremas um para mostrar que os genes com grau de adaptação nulo satisfazem ao lema, e outro para os genes com grau de adaptação diferente de zero.

1. $Teorema$
   $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), grauI(gI_\lambda)) \in maior\mathbb{Q}$
2. $GrauIInocuoDef$
   $\forall g : GI \mid g \neq gI_\lambda \bullet (grauI(g), (0,1)) \in maior\mathbb{Q}$
3. $GrauIUmDef$
   $\forall g : GI \mid g \neq gI_\lambda \bullet ((1,1), (0,1)) \in maior\mathbb{Q}$
4. $MaiorRacionalDefTrue$.□

Com os mesmos passos simplesmente trocando a propriedade $GrauIUmDef$ pela propriedade $GrauIDoisDef$, mostra-se que o predicado é verdadeiro.

D.2.2 Mesma Característica

Lema D.2.2. $\forall g : GI \bullet (g, g) \in mesmaI$

Prova:

1. $Teorema$
   $\forall g : GI \bullet (g, g) \in mesmaI$
2. $mesmaIDef[g_1 := g, g_2 := g]$
   $\forall g : GI \mid (g, g) \in mesmaI \bullet \exists G_1 : atributoRelevanteI \bullet g \in G_1$
3. $atributoRelevanteIDef[g_1 := g, g_2 := g]$
   $True$.□

Lema D.2.3. $\forall g_1, g_2 : GI \mid (g_1, g_2) \in mesmaI \bullet (g_2, g_1) \in mesmaI$

Prova:

1. $Teorema$
   $\forall g_1, g_2 : GI \mid (g_1, g_2) \in mesmaI \bullet (g_2, g_1) \in mesmaI$

2. $mesmaIDef$
   $\forall g_1, g_2 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \bullet (g_2, g_1) \in mesmaI$

3. $mesmaIDef[g_1 := g_2, g_2 := g_1]$;
   $\forall g_1, g_2 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \bullet$
   $\exists G_2 : atributoRelevanteI \bullet g_1 \in G_2 \land g_2 \in G_2$

4. $instantiate\ G_2 == G_1$
   $True.\Box$

Lema D.2.4. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$

Prova:

1. $Teorema$
   $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$

2. $mesmaIDef$
   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (g_2, g_3) \in mesmaI \bullet$
   $(g_1, g_3) \in mesmaI$

3. $atributoRelevanteIDef$
   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2) \Leftrightarrow$
   $(\exists G_2 : atributoRelevanteI \bullet g_1 \in G_2 \land g_2 \in G_2)) \land (g_2, g_3) \in mesmaI \bullet$
   $(g_1, g_3) \in mesmaI$

4. $instantiate\, G_2 == G_1$;
   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land$
   $(raiz(g_1) = raiz(g_2)) \land (g_2, g_3) \in mesmaI \bullet (g_1, g_3) \in mesmaI$

5. $mesmaIDef[g_1 := g_2, g_2 := g_3]$
   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
   $G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \bullet (g_1, g_3) \in mesmaI$

6. $atributoRelevanteIDef[g_1 := g_2, g_2 := g_3]$
   $\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$

$G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \land$
$(raiz(g_2) = raiz(g_3) \Leftrightarrow (\exists G_3 : atributoRelevanteI \bullet g_2 \in G_3 \land g_3 \in G_3)) \bullet (g_1, g_3) \in mesmaI$

7. $instantiate\, G_3 == G_2;$
$\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
$G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \bullet (g_1, g_3) \in mesmaI$

8. $mesmaIDef[g_2 := g_3]$
$\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
$G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \bullet$
$(\exists G_4 : atributoRelevanteI \bullet g_1 \in G_4 \land g_3 \in G_4)$

9. $atributoRelevanteIDef[g_2 := g_3]$
$\forall g_1, g_2, g_3 : GI \mid G_1 \in atributoRelevanteI \land g_1 \in G_1 \land g_2 \in G_1 \land (raiz(g_1) = raiz(g_2)) \land$
$G_2 \in atributoRelevanteI \land g_2 \in G_2 \land g_3 \in G_2 \land (raiz(g_2) = raiz(g_3)) \land$
$G_3 \in atributoRelevanteI \land g_1 \in G_3 \land g_3 \in G_3 \land (raiz(g_1) = raiz(g_3)) \bullet$
$(\exists G_4 : atributoRelevanteI \bullet g_1 \in G_4 \land g_3 \in G_4)$

10. $instantiate G_4 == G_3$
$True.\square$

D.2.3 Gene Dominante

Lema D.2.5. $\forall g : GI \mid g = gI_\lambda \bullet domiI(g, g) = g \Leftrightarrow (g, g) \in mesmaI$

Prova: Esta prova será dividida em duas partes, na primeira mostra-se que a reflexividade da função $domiI$ é verdade quando $g = gI_\lambda$ e a segunda para o caso contrário.

1. $Teorema$
$\forall g : GI \mid g = gI_\lambda \bullet domiI(g, g) = g \Leftrightarrow (g, g) \in mesmaI$

2. $mesmaIReflexiva[g := gI_\lambda]$
$(gI_\lambda, gI_\lambda) \in mesmaI \bullet domiI(gI_\lambda, gI_\lambda) = gI_\lambda$

3. $DomiIIgualDef[g_1 := gI_\lambda, g_2 := gI_\lambda]$
$(gI_\lambda, gI_\lambda) \in mesmaI \land (domiI(gI_\lambda, gI_\lambda) = gI_\lambda \Leftrightarrow (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}) \bullet$
$(grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}$

4. $MesmoRacionalDef[x := grauI(gI_\lambda), y := grauI(gI_\lambda)]$
   $(gI_\lambda, gI_\lambda) \in mesmaI \wedge (domiI(gI_\lambda, gI_\lambda) = gI_\lambda \Leftrightarrow (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}) \wedge$
   $((\exists p, q : \mathbb{Z} \mid q > 0 \bullet grauI(gI_\lambda) = (p, q))) \bullet (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}$

5. $GrauIInocuoDef$
   $(gI_\lambda, gI_\lambda) \in mesmaI \wedge (domiI(gI_\lambda, gI_\lambda) = gI_\lambda \Leftrightarrow (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}) \wedge$
   $((\exists p, q : \mathbb{Z} \mid q > 0 \bullet grauI(gI_\lambda) = (p, q))) \wedge grauI(gI_\lambda) = (0, 1) \bullet$
   $(grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}$

6. $instantiate\ p == 0$
   $(gI_\lambda, gI_\lambda) \in mesmaI \wedge (domiI(gI_\lambda, gI_\lambda) = gI_\lambda \Leftrightarrow (grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}) \wedge$
   $((\exists q : \mathbb{Z} \mid q > 0 \bullet grauI(gI_\lambda) = (0, q))) \wedge grauI(gI_\lambda) = (0, 1) \bullet$
   $(grauI(gI_\lambda), grauI(gI_\lambda)) \in mesmo\mathbb{Q}$

7. $instantiate\ q == 1$
   $True$

A prova de que a função $domiI$ preserva a propriedade reflexiva para o caso de $g \neq gI_\lambda$ é análoga a prova acima, desdes que se utilise o teorema $GrauINaoInocuoDef$ no quinto passo desta prova com a correspondente instanciação das variáveis p e q. □

Lema D.2.6. $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \wedge (g_1, g_3) \in mesmaI \wedge (g_2, g_3) \in mesmaI \bullet$
$domiI(g_1, domiI(g_2, g_3)) = domiI(domiI(g_1, g_2), g_3)$

Prova: A prova desta propriedade requer a avaliação das relações existentes entre o grau de adaptação dos genes envolvidos, que pode ser:

- $(grauI(g_1), grauI(g_2)) \in mesmo\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q}$;
- $(grauI(g_1), grauI(g_2)) \in menor\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q}$;
- $(grauI(g_1), grauI(g_2)) \in maior\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q}$;
- $(grauI(g_1), grauI(g_2)) \in menor\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in menor\mathbb{Q};\ e$
- $(grauI(g_1), grauI(g_2)) \in maior\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in maior\mathbb{Q}.$

No caso em que o grau de adptação de todos genes envolvidos é o mesmo temos:

1. $Teorema$

   $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \wedge (g_1, g_3) \in mesmaI \wedge (g_2, g_3) \in mesmaI \wedge$
   $(grauI(g_1), grauI(g_2)) \in mesmo\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(grauI(g_1), grauI(g_3)) \in mesmo\mathbb{Q} \bullet domiI(g_1, domiI(g_2, g_3)) = domiI(domiI(g_1, g_2), g_3)$

2. $DomiIIgualDef[g_1 := g_2, g_2 := g_3]$

   $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \wedge (g_1, g_3) \in mesmaI \wedge (g_2, g_3) \in mesmaI \wedge$
   $(grauI(g_1), grauI(g_2)) \in mesmo\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(grauI(g_1), grauI(g_3)) \in mesmo\mathbb{Q} \wedge domiI(g_2, g_3) = g_2 \bullet domiI(g_1, g_2) = domiI(domiI(g_1, g_2), g_3)$

3. $DomiIIgualDef$

   $\forall g_1, g_2, g_3 : GI \mid (g_1, g_2) \in mesmaI \wedge (g_1, g_3) \in mesmaI \wedge (g_2, g_3) \in mesmaI \wedge$
   $(grauI(g_1), grauI(g_2)) \in mesmo\mathbb{Q} \wedge (grauI(g_2), grauI(g_3)) \in mesmo\mathbb{Q} \wedge$
   $(grauI(g_1), grauI(g_3)) \in mesmo\mathbb{Q} \wedge domiI(g_2, g_3) = g_2 \wedge domiI(g_1, g_2) = g_1 \bullet g_1 = domiI(g_1, g_3)$

4. $DomiIIgualDef[g_2 := g_3]$

   $True$

Seguindo-se o raciocíneo acima exposto prova-se que a função $domiI$ é associativa. □

D.2.4 Fecundação

Lema D.2.7. $\forall c : CI \bullet (fecI(c, c), c) \in\equiv I_C$

Prova: A prova é identica a apresentada para o **Lema A.4.12**, desde que resguarada a instanciação dos dados e das funções envolvidas neste lema aos dados e funções apresentadas no Capítulo 5.□

D.3 Problema de Monitoramento de Sinais

D.3.1 Grau de Adaptação do Gene

Lema D.3.1. *(Propriedade Requerida sobre o Gene Inócuo) -*

$\forall g : G \mid g \neq g_\lambda \bullet grau(g), grau(g_\lambda) \in maior\mathbb{Q}$

Prova: Idem Lema 8.2.1 considerando que E é um subconjunto do conjunto $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_\lambda\}$, com cardinalidade igual à 3.□

D.3.2 Mesma Característica

Lema D.3.2. *(A Relação Mesma Característica é Reflexiva)* -

$\forall g : G \bullet (g, g) \in mesma$

Prova:

1. Teorema
   $\forall g : G \bullet (g, g) \in mesma$
2. use mesmaDef [g1 := g , g2 := g]
   $True.\square$

Lema D.3.3. *(A Relação Mesma Característica é Comutativa)* -

$\forall g_1, g_2 : G \mid (g_1, g_2) \in mesma \bullet (g_2, g_1) \in mesma$

Prova:

1. Teorema
   $\forall g_1, g_2 : G \mid (g_1, g_2) \in mesma \bullet (g_2, g_1) \in mesma$
2. use mesmaDef
   $\forall g_1, g_2 : G \mid (\exists G : atributoRelevante \mid \{g_1, g_2\} \subseteq G) \bullet (g_2, g_1) \in mesma$
3. use mesmaDef [g_1 := g_2 , g_2 := g_1]
   $True.\square$

Lema D.3.4. *(A Relação Mesma Característica é Transitiva)* -

$\forall g_1, g_2, g_3 : G \mid (g_1, g_2) \in mesma \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$

Prova:

1. Teorema
   $\forall g_1, g_2, g_3 : G \mid (g_1, g_2) \in mesma \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$
2. use mesmaDef
   $\forall g_1, g_2, g_3 : G \mid (\exists G : atributoRelevante \mid \{g_1, g_2\} \subseteq G) \land (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$
3. use mesmaDef [g_1 := g_2 , g_2 := g_3]
   $\forall g_1, g_2, g_3 : G \mid (\exists G : atributoRelevante \mid \{g_1, g_2\} \subseteq G) \land$
   $(\exists G : atributoRelevante \mid \{g_2, g_3\} \subseteq G) \bullet (g_1, g_3) \in mesma$

4. use mesmaDef [g_2 := g_3]
   $True$.□

D.3.3 Gene Dominante

Lema D.3.5. *(A Função Gene Dominanate é Reflexiva)* - $\forall g : G \bullet domi(g, g) = g$

Prova: Idem Lema 8.4.1.□

Lema D.3.6. *(A Função Gene Dominante é Associativa)* - $\forall g_1, g_2, g_3 : G \bullet$
$domi(g_1, domi(g_2, g_3)) = domi(domi(g_1, g_2), g_3)$

Prova:Idem Lema 8.4.2.□

D.3.4 Fecundação

Lema D.3.7. *(A Função Fecunação é Reflexiva)* - $\forall c : CROMOSSOMO \bullet (fec(c, c), c) \in \equiv_C$

Prova: Idem Lema 8.6.1.□

D.3.5 Troca

Lema D.3.8. *(A Função Troca pode ser Definida como a Composição das Funções Inserção e Supressão)* - $\forall c : CROMOSSOMO;\ G_1, G_2 : \mathbb{P}(G) \bullet troc(c, G_1, G_2) = del(ins(c, G_1), G_2)$

Prova: Idem Lema 8.6.1.□

D.4 Refinamento

Como um mapeamento que formaliza a instanciação entre os tipos e a consequente caracterização da função $Retrievi$ como a identidade, modulo este mapeamento, as provas de refinamento tornam-se bastante simples, já que a estrutura de cada elemento Iestado e operação) do GAADT é preservado pela instanciação.

Lema D.4.1. *O teorema de inicialização é verdadeiro para os sistemas $GAADT$ e $GAADTI$*

Prova:

1. Teorema
   $InitGAADTI \wedge Retrievi' \implies InitGAADT$

2. use $CorteDef[p := pInicialI]$

   $StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
   $MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$
   $StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
   $MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
   $pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
   $mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\}$
   $\implies MACHO' = sel(pcorte(pInicialI), MI) \wedge$
   $FEMEA' = sel(pcorte(pInicialI), FI)$

3. use $adaptVazioDef$

   $StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
   $MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$
   $StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
   $MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
   $pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
   $mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\} \wedge adapt(\emptyset) = (0, 1)$
   $\implies MACHO' = sel(pcorte(pInicialI), MI) \wedge$
   $FEMEA' = sel(pcorte(pInicialI), FI)$

4. use $adaptDef$

   $StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
   $MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$
   $StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
   $MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
   $pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
   $mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\} \wedge$
   $adapt(\emptyset) = (0, 1) \wedge adapt(c) = adaptRecursivo(c, c)$
   $\implies MACHO' = sel(pcorte(pInicialI), MI) \wedge FEMEA' = sel(pcorte(pInicialI), FI)$

5. use $CorteIDef[pI := pInicial]$

   $StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
   $MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$
   $StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
   $MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$

$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0, 1) \wedge adapt(c) = adaptRecursivo(c, c)$
$\implies MACHO' = sel(pcorte(pInicialI)\}, MI) \wedge FEMEA' = sel(pcorte(pInicialI), FI)$

6. use $adaptIVazioDef$

$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$
$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0, 1) \wedge adaptI(\emptyset) = (0, 1) \wedge adapt(c) = adaptRecursivo(c, c)$
$\implies MACHO' = sel(pcorte(pInicialI), MI) \wedge FEMEA' = sel(pcorte(pInicialI), FI)$

7. use $adaptIDef$

$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$
$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0, 1) \wedge adaptI(\emptyset) = (0, 1)$
$\wedge adapt(c) = adaptRecursivo(c, c) \wedge adaptI(c) = adaptRecursivoI(c, c)$
$\implies MACHO' = sel(pcorte(pInicialI), MI) \wedge FEMEA' = selpcorte(pInicialI), FI)$

8. use $adaptRecursivoFimDef$

$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$

$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0, 1) \wedge adaptI(\emptyset) = (0, 1) \wedge adapt(c) = adaptRecursivo(c, c) \wedge$
$adaptI(c) = adaptRecursivoI(c, c) \wedge adaptRecursivo(0, \emptyset) = (0, 1)$
$\implies MACHO' = selpcorte(pInicialI), MI) \wedge FEMEA' = selpcorte(pInicialI), FI)$

9. use $adaptRecursivoDef[c_1 := c, c_2 := c]$
$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$
$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0, 1) \wedge adaptI(\emptyset) = (0, 1) \wedge$
$adapt(c) = adaptRecursivo(c, c) \wedge adaptI(c) = adaptRecursivoI(c, c) \wedge$
$adaptRecursivo(0, \emptyset) = (0, 1) \wedge adaptRecursivo(c_1, c_2) = adicao\mathbb{Q}(adaptRecursivo(c_1, c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta(c_1, g), grau(g)))$
$\implies MACHO' = selpcorte(pInicialI), MI) \wedge FEMEA' = selpcorte(pInicialI), FI)$

10. use $adaptRecursivoIFimDef[c_1 := c, c_2 := c]$
$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = selI(pcorteI(pInicialI), MI) \wedge FEMEAI' = selI(pcorteI(pInicialI), FI) \wedge$
$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c), adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c), adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c), adaptMediaI(pInicialI))\} \wedge$

$adapt(\emptyset) = (0,1) \wedge adaptI(\emptyset) = (0,1) \wedge$
$adapt(c) = adaptRecursivo(c,c) \wedge adaptRecursivo(0,\emptyset) = (0,1) \wedge$
$adaptRecursivoI(0,\emptyset) = (0,1) \wedge$
$adaptRecursivo(c_1,c_2) = adicao\mathbb{Q}(adaptRecursivo(c_1,c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta(c_1,g),grau(g)))$
$\Longrightarrow MACHO' = selpcorte(pInicialI),MI) \wedge FEMEA' = selpcorte(pInicialI),FI)$

11. use $adaptRecursivoIDef[c_1 := c, c_2 := c]$
$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = selI(pcorteI(pInicialI),MI) \wedge FEMEAI' = selI(pcorteI(pInicialI),FI) \wedge$
$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c),adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c),adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c),adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c),adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0,1) \wedge adaptI(\emptyset) = (0,1) \wedge adapt(c) = adaptRecursivo(c,c) \wedge$
$adaptRecursivo(0,\emptyset) = (0,1) \wedge adaptRecursivoI(0,\emptyset) = (0,1) \wedge$
$adaptRecursivo(c_1,c_2) = adicao\mathbb{Q}(adaptRecursivo(c_1,c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta heta(c_1,g),grau(g))) \wedge$
$adaptRecursivoI(c_1,c_2) = adicao\mathbb{Q}(adaptRecursivoI(c_1,c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta I(c_1,g),grauI(g)))$
$\Longrightarrow MACHO' = selpcorte(pInicialI),MI) \wedge FEMEA' = sel(pcorte(pInicialI),FI)$

12. use $SelDef[p := pInicialI, r := MI]$
$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = selI(pcorteI(pInicialI),MI) \wedge FEMEAI' = selI(pcorteI(pInicialI),FI) \wedge$
$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c),adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c),adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c),adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c),adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0,1) \wedge adaptI(\emptyset) = (0,1) \wedge adapt(c) = adaptRecursivo(c,c) \wedge$

$adaptRecursivo(0,\emptyset) = (0,1) \wedge adaptRecursivoI(0,\emptyset) = (0,1) \wedge$
$adaptRecursivo(c_1,c_2) = adicao\mathbb{Q}(adaptRecursivo(c_1,c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta(c_1,g),grau(g))) \wedge$
$adaptRecursivoI(c_1,c_2) = adicao\mathbb{Q}(adaptRecursivoI(c_1,c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta I(c_1,g),grauI(g)))$
$\Longrightarrow MACHO' = pcorte(pInicialI) \cap MI \wedge FEMEA' = sel(pcorte(pInicialI),FI)$

13. use $SelDef[p := pInicialI, r := FI]$
$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = selI(pcorteI(pInicialI),MI) \wedge FEMEAI' = selI(pcorteI(pInicialI),FI) \wedge$
$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c),adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c),adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c),adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c),adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0,1) \wedge adaptI(\emptyset) = (0,1) \wedge adapt(c) = adaptRecursivo(c,c) \wedge$
$adaptRecursivo(0,\emptyset) = (0,1) \wedge adaptRecursivoI(0,\emptyset) = (0,1) \wedge$
$adaptRecursivo(c_1,c_2) = adicao\mathbb{Q}(adaptRecursivo(c_1,c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta(c_1,g),grau(g))) \wedge$
$adaptRecursivoI(c_1,c_2) = adicao\mathbb{Q}(adaptRecursivoI(c_1,c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta I(c_1,g),grauI(g)))$
$\Longrightarrow MACHO' = pcorte(pInicialI) \cap MI \wedge FEMEA' = pcorte(pInicialI) \cap FI$

14. use $SelIDef[p := pInicialI, r := MI]$
$StateGAADTI' \wedge TxI' = \{\} \wedge pAtualI' = pInicialI \wedge pcruzI' = \{\} \wedge pmutI' = \{\} \wedge tI' = 0 \wedge$
$MACHOI' = pcorteI(pInicialI) \cap MI \wedge FEMEAI' = selI(pcorteI(pInicialI),FI) \wedge$
$StateGAADT' \wedge Tx' = TxI' \wedge pAtual' = pAtualI' \wedge pcruz' = pcruzI' \wedge pmut' = pmutI' \wedge t' = tI' \wedge$
$MACHO' = MACHOI' \wedge FEMEA' = FEMEAI' \wedge$
$pcorte(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adapt(c),adaptMedia(pInicialI)) \vee$
$mesmo\mathbb{Q}(adapt(c),adaptMedia(pInicialI))\} \wedge$
$pcorteI(pInicialI) = \{c : pInicialI \mid maior\mathbb{Q}(adaptI(c),adaptMediaI(pInicialI)) \vee$
$mesmo\mathbb{Q}(adaptI(c),adaptMediaI(pInicialI))\} \wedge$
$adapt(\emptyset) = (0,1) \wedge adaptI(\emptyset) = (0,1) \wedge adapt(c) = adaptRecursivo(c,c) \wedge$

$adaptRecursivo(0, \emptyset) = (0, 1) \wedge adaptRecursivoI(0, \emptyset) = (0, 1) \wedge$
$adaptRecursivo(c_1, c_2) = adicao\mathbb{Q}(adaptRecursivo(c_1, c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta(c_1, g), grau(g))) \wedge$
$adaptRecursivoI(c_1, c_2) = adicao\mathbb{Q}(adaptRecursivoI(c_1, c_2 \setminus \{g\}),$
$multiplicacao\mathbb{Q}(\Theta I(c_1, g), grauI(g)))$
$\implies FEMEA' = pcorte(pInicialI) \cap FI$

15. use $SelIDef[p := pInicialI, r := FI]$
    $True$.□

Lema D.4.2. *$TerminaPorAdaptaoI$ é um refinamento de $TerminaPorAdaptao$*

Prova: Pela Definição 5.1.2 para provar que $TerminaPorAdaptacaoI$ é um refinamento válido de $TerminaPorAdaptacao$, tem-se que provar que os teoremas de inicialização, de aplicabilidade e de corretude entre estes dois esquemas são válidos para os estados dos sistemas $GAADTI$ e $GAADT$.

O teorema de inicialização é válido pelo Lema D.4.1. A prova da validade do teorema de aplicabilidade é apresentada abaixo.

1. Teorema
   pre $TerminaAdaptacao \wedge Retrievi \implies$ pre $TerminaAdaptacaoI$

2. reduzindo
   $True$

A prova do teorema de corretude é:

1. Teorema
   pre $TerminaAdaptacao \wedge Retrievi \wedge TerminaAdaptacaoI \wedge Retrievi' \implies TerminaAdaptacao$

2. reduzindo
   $True$.□

Lema D.4.3. *$TerminaPorTempoI$ é um refinamento de $TerminaPorTempo$*

Prova: Pela Definição 5.1.2 para provar que $TerminaPorTempoI$ é um refinamento de $TerminaPorTempo$, tem-se que provar que o teorema de inicialização, de aplicabilidade e de corretude entre estes dois esquemas são válidos para os estados dos modelos $GAADTI$ e $GAADT$.

A prova do teorema de inicialização foi apresentada no Lema D.4.1. A prova da validade do teorema de aplicabilidade é obtida através dos seguintes passos:

1. Teorema

   pre $TerminaTempo \wedge Retrievi \implies$ pre $TerminaTempoI$

2. atribuindo a T o valor TI

   $True$

A provar o teorema de corretude é:

1. Teorema

   pre $TerminaTempo \wedge Retrievi \wedge TerminaTempoI \wedge Retrievi' \implies TerminaTempo$

2. reduzindo

   $True$.□

Lema D.4.4. *$TerminaI$ é um refinamento de $Termina$*

Prova: A Definição 5.5.4 estabelece quatro condições necessárias para que uma operação concreta descrita em função da disjunção de duas outras operações possa refina a sua versão abstrata. As duas primeiras condições exigem que cada operação concreta participante desta disjunção refine a sua versão abstrata, o que já foi provado nos Lemas D.4.2 e D.4.3.

A terceira condição exige que o esquema $TerminaPorAdaptacao \vee TerminaPorTempo$ seja consistente com o esquema pre $TerminaPorAdaptacao \wedge TerminaPorTempoI$.

1. Teorema

   $\forall Retrievi \wedge Retrievi' \wedge$ pre $TerminaPorAdaptacao \wedge TerminaPorTempoI \implies$
   $TerminaPorAdaptacao \vee TerminaPorTempo$,

2. reduzindo

   $True$

Finalmente a quarta condição exige que o esquema $TerminaPorTempo \vee TerminaPorAdaptacao$ seja consistente com o esquema pre $TerminaPorTempo \wedge TerminaPorAdaptacaoI$, cuja prova é exibida abaixo:

1. Teorema

   $Retrievi \wedge Retrievi' \wedge$ pre $TerminaPorTempo \wedge TerminaPorAdaptacaoI \implies$
   $TerminaPorTempo \vee TerminaPorAdaptacao$

2. atribui a k o valor kI, e a T o valor TI

   $Retrievi \wedge Retrievi' \wedge$ pre $TerminaPorTempo \wedge TerminaPorAdaptacaoI \wedge k = kI \implies$
   $TerminaPorTempo \vee TerminaPorAdaptacao$

3. atribua a função $grau$ a faunção $grauI$ e a função Θ a função ΘI
   $Retrievi \wedge Retrievi' \wedge$ pre $TerminaPorTempo \wedge TerminaPorAdaptacaoI \wedge$
   $k = kI \wedge (\forall c_1 : p_{atual}I;\ c_2 : p_{atual} \mid c_1 = c_2 \bullet adapt(c_2) = adaptI(c_1)) \implies$
   $TerminaPorTempo \vee TerminaPorAdaptacao$

4. substituindo c_1 e c_2 do teorema anterior por c
   $True.\Box$

Lema D.4.5. *$CruzamentoI$ é um refinamento de $Cruzamento$*

Prova: Pela Definição 5.1.2 para provar que $CruzamentoI$ é um refinamento válido de $Cruzamento$, tem-se que provar que o teorema de inicialização, de aplicabilidade e de corretude entre estas duas operações são válidos para os estados dos sistemas $GAADTI$ e $GAADT$.

A prova do teorema de inicialização foi apresentada no Lema D.4.1. O teorema de aplicabilidade é válido como mostrado abaixo.

1. Teorema
   pre $Cruzamneto \wedge Retrievi \implies CruzamentoI$

2. reduzindo
   $True$

A provar o teorema de corretude consiste dos seguintes passos::

1. Teorema
   $SatetGAADT \wedge StateGAADTI \wedge StateGAADTI' \wedge Retrievi \wedge Retrievi' P_{atual} \in P \wedge$
   $(\exists P'_{cruz}/0 : P \bullet P'_{cruz}/0 = \{c_1 : C \mid (\exists c_2 : MACHO;\ c_3 : FEMEA \bullet c_1 \in cruz(c_2, c_3)) \wedge$
   $(\forall c_4, c_5 : CI \mid (c_4, c_5) \in Tx \bullet (c_4, c_1) \not\equiv C \wedge (c_5, c_1) \not\equiv C)\}) \wedge$
   $PI'_{cruz} = \{c_1 : CI \mid (\exists c_2 : MACHOI;\ c_3 : FEMEAI \bullet c_1 \in cruzI(c_2, c_3)) \wedge$
   $(\forall c_4, c_5 : CI \mid (c_4, c_5) \in TxI \bullet (c_4, c_1) \not\equiv I_C \wedge (c_5, c_1) \not\equiv I_C)\}$
   $\implies P'_{cruz} = \{c_1 : C \mid (\exists c_2 : MACHO;\ c_3 : FEMEA \bullet c_1 \in cruz(c_2, c_3)) \wedge$
   $(\forall c_4, c_5 : C \mid (c_4, c_5) \in Tx \bullet (c_4, c_1) \not\equiv C \wedge (c_5, c_1) \not\equiv C)\}$

2. substituindo $P'_{cruz}/0$ por P'_{cruz} no teorema acima
   $True.\Box$

Lema D.4.6. *$MutacaoI$ é um refinamento de $Mutacao$*

Prova: Pela Definição 5.1.2 para provar que $MutacaoI$ é um refinamento válido de $Mutacao$, tem-se que provar que o teorema de inicialização, de aplicabilidade e de corretude entre estas duas operações são válidos para os estados dos sistemas $GAADTI$ e $GAADT$.

A prova do teorema de inicialização foi apresentada no Lema D.4.1. A prova da validade do teorema de aplicabilidade é dada pelos seguintes passos:

1. Teorema

   pre $Mutacao \wedge Retrievi \implies$ pre $MutacaoI$

2. reduzindo

   $True$

A prova do teorema de corretude é apresentada a seguir:

1. Teorema

   $SatetGAADT \wedge StateGAADTI \wedge StateGAADTI' \wedge Retrievi \wedge Retrievi'$
   $(\exists P'_{mut}/0 : P \bullet P'_{mut}/0 = \{c_1 : C \mid (\forall c_2 : P_{atual} \setminus p_{corte}(P_{atual}) \bullet c_1 \in mut(c_2)) \wedge$
   $(\exists c_3, c_4 : C \mid (c_3, c_4) \in Tx \bullet (c_1, c_3) \notin\equiv C \wedge (c_1, c_4) \notin\equiv C)\} \wedge$
   $PI'_{mut} = \{c_1 : CI \mid (\forall c_2 : PI_{atual} \setminus pI_{corte}(PI_{atual}) \bullet c_1 \in mutI(c_2)) \wedge$
   $(\exists c_3, c_4 : CI \mid (c_3, c_4) \in TxI \bullet (c_1, c_3) \notin\equiv I_C \wedge (c_1, c_4) \notin\equiv I_C)\}$
   $\implies P'_{mut} = \{c_1 : C \mid (\forall c_2 : P_{atual} \setminus p_{corte}(P_{atual}) \bullet c_1 \in mut(c_2)) \wedge$
   $(\exists c_3, c_4 : C \mid (c_3, c_4) \in Tx \bullet (c_1, c_3) \notin\equiv C \wedge (c_1, c_4) \notin\equiv C)\}$

2. substituindo $P'_{mut}/0$ por P'_{mut} no teorema acima

   $True$.□

Lema D.4.7. *$CruzamentoI \wedge MutacaoI$ é um refinamento de $Cruzamento \wedge Mutacao$*

Prova: A Definição 5.5.3 estabelece três condições necessárias para que uma operação concreta descrita em função da conjunção de duas outras operações possa refina a sua versão abstrata. As duas primeiras condições exigem que cada operação concreta participante desta conjunção refine a sua versão abstrata, o que já foi provado pelos Lemas D.4.5 e D.4.6.

A terceira condição exige que o esquema $Cruzamento \wedge Mutacao$ seja consistente com o esquema $CruzamentoI \wedge MutacaoI$ o que é válido, como apresentado abaixo:

1. Teorema

   $\forall Retrievi \wedge Retrievi' \wedge$ pre $(Cruzamento \wedge Mutacao) \wedge$ pre $(Cruza- mentoI \wedge MutacaoI) \implies$
   pre $((Cruzamento \wedge CruzamentoI) \wedge (Mutacao \wedge MutacaoI))$

2. reduzindo

   $True.\square$

Lema D.4.8. *$IteraoI$ é um refinamento de $Iterao$*

Prova: Pela Definição 5.1.2 para provar que $MutacaoI$ é um refinamento aceitável de $Mutacao$, tem-se que provar que o teorema de inicialização, de aplicabilidade e de corretude entre estas duas operações são válidos para os estados dos sistemas $GAADTI$ e $GAADT$.

A prova do teorema de inicialização foi apresentada no Lema D.4.1 A prova do teorema de aplicabilidade é válida, como mostrado abaixo:

1. Teorema

   pre $Iteracao \wedge Retrievi \implies$ pre $IteracaoI$

2. reduzindo

   $True$

Enquanto a do teorema de corretude pre $Iteracao \wedge Retrievi \wedge IteracaoI \wedge Retrievi' \implies Iteracao$ é apresentada a seguir:

1. Teorema

   $SatetGAADT \wedge StateGAADTI \wedge StateGAADTI' \wedge Retrievi \wedge Retrievi' \wedge$
   $(\exists Tx'/0 : C \leftrightarrow C \bullet Tx'/0 = Tx \cup \{c_1 : p_{atual};\ c_2 : p_{cruz} \mid \exists c_3 : p_{atual} \bullet c_2 \subseteq cruz(c_1, c_2)\}$
   $\cup \{c_1 : p_{atual};\ c_2 : p_{mut} \mid c_2 \subseteq mut(c_1)\}) \wedge$
   $(\exists P'_{atual}/0 : P \bullet P'_{atual}/0 = p_{corte}(P_{atual}) \cup P_{cruz} \cup P_{mut}) \wedge$
   $TxI' = TxI \cup \{c_1 : pI_{atual};\ c_2 : pI_{cruz} \mid \exists c_3 : pI_{atual} \bullet c_2 \subseteq cruzI(c_1, c_2)\}$
   $\cup \{c_1 : pI_{atual};\ c_2 : pI_{mut} \mid c_2 \subseteq mutI(c_1)\} \wedge$
   $PI'_{atual} = pI_{corte}(PI_{atual}) \cup PI_{cruz} \cup PI_{mut}$
   $\implies Tx' = Tx \cup \{c_1 : p_{atual};\ c_2 : p_{cruz} \mid \exists c_3 : p_{atual} \bullet c_2 \subseteq cruz(c_1, c_2)\}$
   $\cup \{c_1 : p_{atual};\ c_2 : p_{mut} \mid c_2 \subseteq mut(c_1)\}) \wedge$
   $P'_{atual} = p_{corte}(P_{atual}) \cup P_{cruz} \cup P_{mut}$

2. substituindo $P'_{atual}/0$ por P'_{atual} no teorema acima

   $True.\square$

Lema D.4.9. $(CruzamentoI \wedge MutacaoI)\ {}^{0}_{9}\ IteracaoI$ *é um refinamento de* $(Cruzamento \wedge Mutacao)\ {}^{0}_{9}$ $Iteracao$

Prova: A Definição 5.5.6 estabelece três condições necessárias para que uma operação concreta descrita em função da composição de duas outras operações possa refina a sua versão abstrata. As duas primeiras condições exigem que cada operação concreta participante desta composição refine a sua versão abstrata, o que já foi provado pelos Lemas D.4.7 e D.4.8.

A terceira condição exige que o esquema $(Cruzamento \wedge Mutacao)\ {}^{0}_{9}\ Iteracao$ seja consistente com o esquema $(CruzamentoI \wedge MutacaoI)\ {}^{0}_{9}\ IteracaoI$, o que é válido pelos seguintes passos:

1. Teorema
   $\exists StateGAADT''; \ StateGAADTI'' \bullet \forall Retrievi \wedge Retrievi' \wedge$ pre $((Cruzamento \wedge Mutacao)$
   $[StatGAADT'StateGAADT''] \wedge Iteracao[StatGAADTStateGAADT'']) \wedge$ pre $((CruzamentoI \wedge$
   $MutacaoI)$
   $[StatGAADTI'StateI''] \wedge IteracaoI[StatGAADTIStateGAADTI'']) \implies$ pre $((Cruzamento \wedge$
   $Mutacao)$
   $[StatGAADT'StateGAADT''] \wedge (CruzamentoI \wedge MutacaoI)[StatGAADTI'StateGAADTI'']) \wedge$
   $(Iteracao[StatGAADTStateGAADT''] \wedge IteracaoI[StatGAADTIStateGAADTI''])$

2. reduzindo
   $True.\square$

Lema D.4.10. *O esquema* $TerminaI((CruzamentoI \wedge MutacaoI)\ {}^{0}_{9}\ IteracaoI)$ *é um refinamento da operação* $Termina((Cruzamento \wedge Mutacao)\ {}^{0}_{9}\ Iteracao)$

Prova: A Definição 5.5.4 estabelece quatro condições necessárias para que uma operação concreta descrita em função da disjunção de duas outras operações possa refina a sua versão abstrata. As duas primeiras condições exigem que cada operação concreta participante desta composição refine a sua versão abstrata, o que já foi provado pelos Lemas D.4.4 e D.4.9.

A terceira condição exige que o esquema $GAADT$ seja consistente com o esquema $GAADTI$, o que é válido pelos seguintes passos:

1. Teorema
   $\forall Retrievi \wedge Retrievi' \wedge$ pre $Termina \wedge (CruzamentoI \vee MutacaoI)\ {}^{0}_{9}\ IteracaoI \implies Termina \vee$
   $(Cruzamento \vee Mutacao)\ {}^{0}_{9}\ Iteracao$

2. reduzindo
   $True$

Finalmente a quarta condição exige que o esquema $(Cruzamento \vee Mutacao) \overset{0}{9} Iteracao \vee TerminaI$ seja consistente com o esquema pre $(Cruzamento \vee Mutacao) \overset{0}{9} Iteracao \wedge TerminaI$, a qual também é verdadeira por definição.