HELDER ZANDONADI MAIA

OTIMIZAÇÃO DA RESPOSTA DA POTÊNCIA ATIVA DE UM INVERSOR CONECTADO À REDE ELÉTRICA USANDO ALGORITMOS GENÉTICOS

UFMS

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

CAMPO GRANDE — MS

2012

OTIMIZAÇÃO DA RESPOSTA DA POTÊNCIA ATIVA DE UM INVERSOR CONECTADO À REDE ELÉTRICA USANDO ALGORITMOS GENÉTICOS

HELDER ZANDONADI MAIA

CAMPO GRANDE

2012

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

OTIMIZAÇÃO DA RESPOSTA DA POTÊNCIA ATIVA DE UM INVERSOR CONECTADO À REDE ELÉTRICA USANDO ALGORITMOS GENÉTICOS

Dissertação submetida à
Universidade Federal de Mato Grosso do Sul
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica

HELDER ZANDONADI MAIA

Campo Grande, Maio de 2012

OTIMIZAÇÃO DA RESPOSTA DA POTÊNCIA ATIVA DE UM INVERSOR CONECTADO À REDE ELÉTRICA USANDO ALGORITMOS GENÉTICOS

Helder Zandonadi Maia

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em *Energia*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Mato Grosso do Sul.'

João Onofre Pereira Pinto, PhD.
Orientador

Luciana Cambraia Leite, Dr[a].
Coordenadora do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

João Onofre Pereira Pinto, PhD.
Presidente

Humberto Mendes Mazzini, Dr.

Jurandir de Oliveira Soares, Dr.

Ruben Barros Godoy, Dr.

Agradecimentos

Aos meus pais e irmã pelo suporte e apoio.

Ao Prof. João Onofre Pereira Pinto por ter ido muito além das obrigações de orientador e, especialmente, pela ajuda que somente um verdadeiro amigo pode prestar.

Ao Prof. Ernane Antônio Alves Coelho, pela disponibilidade para esclarecer nossas dúvidas e realização dos ensaios experimentais.

Aos colegas do BATLAB, pelas oportunidades compartilhadas de aprendizado e crescimento.

Aos professores e funcionários do Programa de Pós-Graduação do Departamento de Engenharia Elétrica, por terem tornado possível a realização desta dissertação.

A todos cujos esforços serviram de base para este trabalho.

Resumo da Dissertação apresentada à UFMS como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

OTIMIZAÇÃO DA RESPOSTA DA POTÊNCIA ATIVA DE UM INVERSOR CONECTADO À REDE ELÉTRICA USANDO ALGORITMOS GENÉTICOS

HELDER ZANDONADI MAIA

Maio/2012

Orientador: João Onofre Pereira Pinto, PhD.

Área de Concentração: Energia.

Palavras-chave: otimização, algoritmos genéticos, conexão à rede, inversores, meta-heurísticas.

Número de Páginas: 89.

Neste trabalho analisa-se o emprego de algoritmos genéticos na otimização da resposta do fluxo de potência ativa de um inversor conectado à rede elétrica. É realizada uma revisão de técnicas de otimização analíticas e heurísticas considerando sua aplicabilidade à metodologia de paralelização baseada em curvas P–ω e Q–V. Também é introduzido um modelo discreto baseado em Espaço de Estados cuja acurácia é confirmada frente ao modelo de Pequenos Sinais e a um *software* de simulação de circuitos. A consistência das respostas obtidas pelos algoritmos genéticos é avaliada através de múltiplas execuções e os melhores e piores resultados verificados experimentalmente. Por fim, discutem-se as causas das divergências entre os resultados de simulação e empírico e medidas que podem ser tomadas para reduzi-las.

Abstract of Dissertation presented to UFMS as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

ACTIVE POWER FLOW OPTIMIZATION OF A GRID CONNECTED INVERTER USING GENETIC ALGORITHMS

HELDER ZANDONADI MAIA

May/2012

Advisor: João Onofre Pereira Pinto, PhD.

Area of Concentration: Energy.

Keywords: optimization, genetic algorithms, grid connection, inverters, metaheuristics.

Number of Pages: 89.

This work considers the application of genetic algorithms to the optimization of the active power flow of a grid connected inverter. It reviews analytic and heuristic optimization techniques according to their suitability to the parallelization methodology based on P–ω and Q–V curves. A discrete state space based model is also introduced and its accuracy in relation to the Small Signals model and results from a circuit simulator are established. The consistency of the genetic algorithms' results is assessed through multiple runs and experimentally verifying the best and worst ones. Finally, the causes of the divergences between simulation and experimental results and possible corrective measures are addressed.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

A expansão do uso de fontes renováveis de energia com capacidade de conexão à rede pública de distribuição de energia elétrica demanda sistemas de controle de paralelismo robustos e expansíveis. Existem duas metodologias básicas de controle de paralelismo para estas unidades: com comunicação entre unidades inversoras e sem comunicação [1].

As metodologias que empregam comunicação no controle do paralelismo conseguem obter uma distribuição mais eficiente do carregamento entre unidades inversoras, no entanto, estão sujeitas a maiores instabilidades em momentos de oscilações de carga e seu tempo de resposta apresenta grande dependência da velocidade de comunicação. A flexibilidade de instalação de unidades deste tipo é limitada pela necessidade de comunicação entre unidades remotas, o que também apresenta efeito adverso sobre a confiabilidade do sistema. Principalmente no contexto de Sistemas de Energia Ininterrupta (UPS — *Uninterruptible Power Supplies*), as limitações previamente mencionadas não são desejadas.

As metodologias sem comunicação no controle possibilitam menores tempos de transferência e períodos transitórios (causados por oscilações nas condições de carga) de menor duração. Dentre elas, será abordado o emprego de curvas P–ω e Q–V [2, 1]. Esta técnica procura resolver o problema do controle de unidades independentes transformando-o num problema já conhecido e solucionado: o controle das máquinas síncronas conectadas ao sistema elétrico.

Apesar do sistema elétrico apresentar muitas máquinas síncronas operando conectadas

não é necessária nenhuma comunicação entre as unidades geradoras. Cada máquina controla independentemente o seu fluxo de potência ativa e reativa.

A própria dinâmica da máquina síncrona é responsável pela redução da velocidade angular ω quando há aumento da solicitação de potência ativa P. Enquanto o AVR (*Automatic Voltage Regulator*) encarrega-se de controlar a tensão V gerada pela máquina de acordo com a potência reativa Q demandada pelo sistema elétrico.

O problema desta abordagem é que sua resposta é dependente dos parâmetros que permitem ao inversor se comportar como uma máquina síncrona: a inclinação das curvas P–ω e Q–V; a frequência de corte do filtro utilizado na medição de potência e o valor do ganho de realimentação de fase. A complexidade da relação entre estes parâmetros e a resposta da potência ativa torna a sua seleção difícil.

Dentre as metodologias de otimização disponíveis para seleção destes parâmetros de controle, os algoritmos genéticos constituem uma técnica robusta, genérica, relativamente simples e amplamente aplicada, justificando a análise do seu emprego.

1.1 Visão Geral de Sistemas UPS Distribuídos

A Figura 1.1 apresenta um sistema distribuído de UPS's do tipo *on-line*. As unidades são caracterizadas por serem do tipo *on-line* pelo fato da rede de segurança sempre ser alimentada através da bateria da UPS. As principais vantagens desta abordagem são a minimização do tempo de transferência e a isolação das cargas na rede de segurança de perturbações na rede elétrica. As maiores desvantagens são decorrentes da dupla conversão de energia necessária: maior custo e redução de eficiência, entre elas.

Independente do tipo de UPS (*on-line* ou *line interactive*), num sistema distribuído surge a necessidade de se optar entre duas filosofias básicas do controle das diferentes unidades. O objetivo de ambas é coordenar o funcionamento das UPS's de maneira a atender as variações

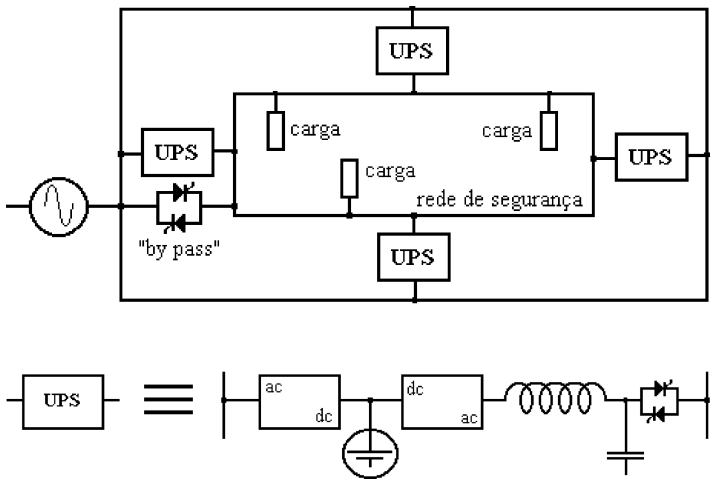

Figura 1.1: Sistema distribuído de UPS's do tipo *on-line*. Fonte: [3].

de demanda das cargas da melhor forma possível.

A filosofia com comunicação *explícita* utiliza canais de comunicação dedicados interligando as várias unidades para alcançar este objetivo. Uma forma como esta abordagem lida com os problemas provocados pela necessidade de ligações de comunicação é através do uso de diferentes topologias em suas redes de controle [3]. Diferentes topologias representam diferentes compromissos entre facilidade de instalação de novas unidades, uso ótimo das UPS's existentes para cada situação de carga e confiabilidade geral do sistema.

A filosofia sem comunicação explícita não necessita de ligações dedicadas para comunicação. A informação necessária para atender a carga já está incorporada no projeto de cada unidade. A sua aplicação explora justamente o fato que cada UPS, quando apresentada com as mesmas variáveis elétricas locais, agirá exatamente da mesma forma. Enquanto esta dependência exclusiva de variáveis locais tornam sistemas sem comunicação mais confiáveis e expansíveis, eles não conseguem atingir a mesma eficiência que é possível em sistemas de controle com comunicação explícita.

Dentre as metodologias não baseadas em comunicação, o controle por curvas P–ω e Q–V encontra-se entre os mais simples e também figura entre os mais empregados [3]. Também existem estratégias híbridas, nas quais a distribuição de carregamento das UPS's é feita

através de comunicação explícita quando os canais de comunicação estão interconectados e funcionando normalmente.

1.2 Otimização de Parâmetros de Controle

1.2.1 Teoria de Controle

As ferramentas fornecidas pela Teoria de Controle para sintonia dos parâmetros de um dado sistema tendem a ser voltadas para a etapa de concepção do controle. Isto é consistente com a base eminentemente matemática desta área e a consequente preocupação com a caracterização precisa de uma determinada técnica.

No entanto, isto também leva à existência de diversas técnicas com diferentes generalizações. Elas tornam possível a determinação de garantias com graus variáveis de força e aplicabilidade. Contudo, mesmo quando uma determinada generalização é aplicável ao problema em estudo, o projetista tem que redefinir o problema em termos compatíveis para usufruir dos benefícios providos pela teoria específica.

Além disso, estão disponíveis técnicas com diversos graus de sofisticação. Desde técnicas clássicas utilizando aproximação para sistemas de segunda ordem ou otimização da norma euclidiana de um sistema linear invariante no tempo, até metodologias de controle robusto empregando normas H_2 ou H_∞ [4]. Normalmente, quanto maior o grau de sofisticação, maior conhecimento específico é exigido do projetista.

Estas técnicas com maior sofisticação dificultam o emprego de uma abordagem de desenvolvimento gradual do sistema de controle (elas não reutilizam os modelos matemáticos mais simples e nem são resultado de uma adaptação do modelo anterior). Especialmente para sistemas de controle concebidos com base em modelos de estabilidade, como é o caso típico de aplicação da Análise de Pequenos Sinais em Sistemas Elétricos de Potência.

Enquanto a referida análise é adequada para o estudo da estabilidade e frutífera para

concepção de novas variantes de controle, normalmente, o modelo não é suficientemente preciso para o emprego de técnicas de controle ótimo.

Adicionalmente, a experiência e intuições acumuladas no modelo de pequenos sinais não são trivialmente transferidas para o contexto de controle ótimo. Já que as métricas sendo otimizadas (as normas) não apresentam uma relação simples com medidas de desempenho clássicas, como o tempo de assentamento.

1.2.2 Teoria de Otimização

Dada as aplicações de sistemas de controle desenvolvidos com base em modelos de estabilidade [3, 5] e a desejabilidade de uma abordagem incremental de projeto, justifica-se a busca de uma alternativa à uma reestruturação radical de um modelo já existente e, eventualmente, operacional.

Diante disso, as ferramentas fornecidas pela Teoria de Otimização mostram-se mais adequadas. Elas permitem ao projetista definir métricas de desempenho derivadas de sua experiência prévia com o sistema. À medida que o entendimento e as necessidades do projeto progridem, seria desejável que tanto o modelo do sistema como métricas de desempenho pudessem ser aprimorados de maneira relativamente simples.

Enquanto isto é alcançável empregando metodologias baseadas em técnicas analíticas, a progressão para modelos mais refinados exige a adoção de técnicas cada vez mais sofisticadas. A alternativa de começar o processo por uma das técnicas mais genéricas também não é satisfatória, pois pode se mostrar desnecessária com a maturação do projeto.

A complexidade e o conhecimento prévio necessários para lidar com otimizações de sistemas com não-linearidades pontuais, também constam entre as dificuldades impostas pelo uso de ferramentas analíticas. Somam-se a elas, a necessidade de formular o problema de modo a atender os requisitos estruturais da técnica. Sejam eles linearidade ou convexidade.

Apesar destas metodologias poderem lidar com restrições aos parâmetros de controle sendo otimizados, elas também necessitam atender certas exigências. Dentre elas diferenciabilidade, impossibilitando a implementação de restrições descontínuas (geralmente, as mais simples de serem formuladas).

O modo como funções multiobjetivos são tratadas é limitado [6, 7] e os resultados fornecidos são pontuais. Este último aspecto é particularmente problemático pelo fato de existirem várias soluções para esta classe de funções, cada uma representando um diferente compromisso entre requerimentos do projeto.

Deste modo, é relevante considerar o emprego de meta-heurísticas. Elas são concebidas para serem flexíveis e ajustáveis para aplicações específicas. Além disso, elas tem sido empregadas com relativo sucesso em diversas áreas onde métodos com base analítica são predominantes [8, 9], incluindo controle de sistemas [10, 11].

No âmbito das meta-heurísticas, algoritmos genéticos apresentam uma literatura abrangendo tanto aspectos teóricos quanto aplicações em diversos domínios [12]. Constituindo uma filosofia amadurecida e que mostrou-se capaz de obter bons resultados sem comprometer os seus princípios básicos.

No contexto de funções multiobjetivo, os algoritmos genéticos (especificamente a vertente NSGA-II) ainda são tidos como o *benchmark* padrão para a avaliação do desempenho de outras heurísticas [13, 14, 7].

2 Modelagem do Sistema de Controle

2.1 Modelo de Pequenos Sinais da Resposta do Ângulo de Carga

Dada a origem do sistema de controle por curvas P–ω e Q–V em Sistemas de Potência, nada mais natural que utilizar os resultados obtidos pela aplicação da análise de pequenos sinais ao sistema composto por um inversor conectado a um barramento infinito. Este sistema é apresentado na Figura 2.1.

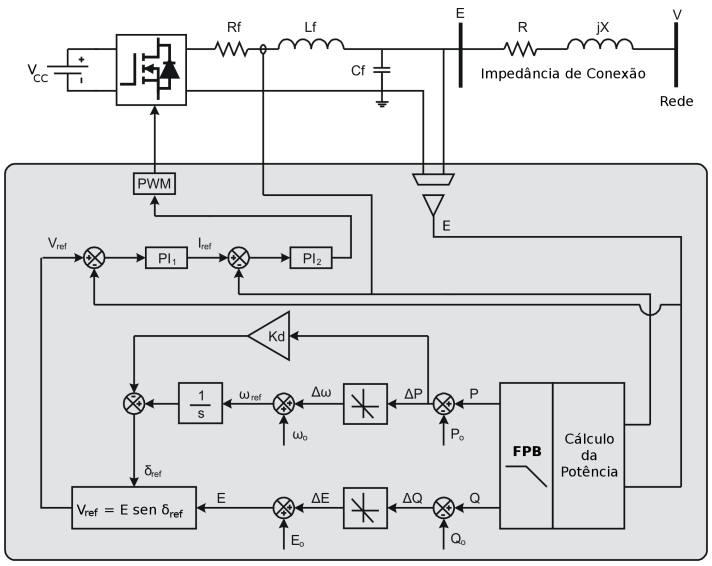

Figura 2.1: Inversor conectado a um barramento infinito através de uma impedância de conexão.

O desenvolvimento do modelo de pequenos sinais para o sistema proposto encontra-se em [15]. Nota-se que o inversor é suposto ser capaz de alterar instantaneamente a amplitude e a fase da sua tensão de saída.

Basicamente, a tensão E sobre o capacitor do filtro de saída do inversor e a corrente total drenada são realimentadas como entrada do sistema de controle. As potências ativa P e reativa Q são obtidas utilizando um filtro passa-baixas com frequência de corte ω_f. As respectivas potências são comparadas com os valores desejados para regime permanente e os respectivos erros são multiplicados pela declividade da respectiva curva.

Deste modo, o erro ΔP é multiplicado pela declividade da curva P–ω (K_p) para que se obtenha a variação de frequência $\Delta\omega$ necessária para se obter a potência desejada em regime permanente. Ela é somada a frequência nominal do sistema, ω_0, e integrada para que se tenha o ângulo de carga.

O ganho K_d permite uma resposta imediata do ângulo de carga do inversor a variações de potência e foi introduzido por [15] para diminuir a magnitude da sobre-resposta da potência ativa presente no controle P–ω convencional. O controle convencional equivale a tornar K_d nulo.

A variação da potência reativa ΔQ ao ser multiplicada pela declividade da curva Q–V (K_v) resulta numa correção ΔE na tensão nominal de saída do inversor (E).

A tensão de referência assim formada passa por controladores PI que geram a modulante para um módulo de chaveamento SPWM (*Sinusoidal Pulse Width Modulation*). A dinâmica destes controladores é desconsiderada na análise de pequenos sinais por ser muito mais rápida que a dinâmica do resto do sistema (devido à indutância de conexão à rede e aos filtros passa-baixas).

O modelo supõe que o sistema opere próximo das condições de regime permanente dadas por uma tensão senoidal de magnitude E_e e fase δ_e na saída do inversor e uma tensão de magnitude V_e e fase nula no barramento infinito. Resultando num fluxo de potência ativa em regime permanente P_e e um fluxo de reativo Q_e.

Assim sendo, a variação do ângulo de carga em torno do valor de regime permanente é

dada por (2.1) e o ângulo de carga por (2.2).

$$\frac{\mathrm{d}^3\Delta\delta(t)}{\mathrm{d}t^3} + a\frac{\mathrm{d}^2\Delta\delta(t)}{\mathrm{d}t^2} + b\frac{\mathrm{d}\Delta\delta(t)}{\mathrm{d}t} + c\Delta\delta(t) = 0 \tag{2.1}$$

$$\delta(t) = \Delta\delta(t) + \delta_e \tag{2.2}$$

As constantes a, b e c em (2.1) são dadas pelas equações (2.3) a (2.5), respectivamente.

$$a = \omega_f(2 + K_v k_{qe}) + K_d\omega_f k_{pd} \tag{2.3}$$

$$b = (1 + K_d k_{pd})(1 + K_v k_{qe})\omega_f^2 + K_p\omega_f k_{pd} - K_d K_v k_{pe} k_{qd}\omega_f^2 \tag{2.4}$$

$$c = K_p\omega_f^2[k_{pd}(1 + K_v k_{qe}) - K_v k_{pe} k_{qd}] \tag{2.5}$$

As constantes k_{pe}, k_{pd}, k_{qe} e k_{qd} são dadas pelas equações (2.6) a (2.9), respectivamente. Nas quais, R e X são, respectivamente, a resistência e a reatância do indutor de conexão à rede elétrica.

$$k_{pe} = \frac{1}{R^2 + X^2}(2RE_e - RV_e + XV_e\operatorname{sen}\delta_e) \tag{2.6}$$

$$k_{pd} = \frac{1}{R^2 + X^2}(RE_eV_e\operatorname{sen}\delta_e + XE_eV_e\cos\delta_e) \tag{2.7}$$

$$k_{qe} = \frac{1}{R^2 + X^2}(2XE_e - XV_e\cos\delta_e - RV_e\operatorname{sen}\delta_e) \tag{2.8}$$

$$k_{qd} = \frac{1}{R^2 + X^2}(XE_eV_e\operatorname{sen}\delta_e - RE_eV_e\cos\delta_e) \tag{2.9}$$

Portanto, para resolver (2.1) é preciso achar as raízes do seu polinômio característico, dado por (2.10).

$$\lambda^3 + a\lambda^2 + b\lambda + c = 0 \tag{2.10}$$

Dependendo das raízes as soluções serão combinações lineares das bases apresentadas na Tabela 2.1. Como o ângulo de carga deve voltar ao valor de equilíbrio quando o tempo tende ao infinito a solução de (2.1) tem de estar sujeita a condição de contorno dada por (2.11). Ou seja, as raízes de (2.10) devem ter partes reais negativas e/ou raiz nula com multiplicidade um.

Tabela 2.1: Soluções de (2.1) em função das raízes de (2.10).

Raízes	Base	Solução ($\Delta\delta(t)$)
$\lambda_i \in \mathbb{R}, \lambda_1 \neq \lambda_2 \neq \lambda_3$	$\{e^{\lambda_1 t}, e^{\lambda_2 t}, e^{\lambda_3 t}\}$	$k_1 e^{\lambda_1 t} + k_2 e^{\lambda_2 t} + k_3 e^{\lambda_3 t}$
$\lambda_i \in \mathbb{R}, \lambda_1 = \lambda_2 = \lambda \neq \lambda_3$	$\{e^{\lambda t}, te^{\lambda t}, e^{\lambda_3 t}\}$	$(k_1 + k_2 t)e^{\lambda t} + k_3 e^{\lambda_3 t}$
$\lambda_i \in \mathbb{R}, \lambda_1 = \lambda_2 = \lambda_3 = \lambda$	$\{e^{\lambda t}, te^{\lambda t}, t^2 e^{\lambda t}\}$	$(k_1 + k_2 t + k_3 t^2)e^{\lambda t}$
$\lambda_1, \lambda_2 \in \mathbb{C}, \lambda_3 \in \mathbb{R}, \lambda_1 = \lambda_2^* = \lambda = \alpha + j\beta$	$\{e^{\lambda t}, e^{\lambda^* t}, e^{\lambda_3 t}\}$	$[(k_1 + k_2)\cos\beta t + j(k_1 - k_2)\text{sen}\,\beta t]e^{\alpha t} + k_3 e^{\lambda_3 t}$

$$\lim_{t\to\infty} \Delta\delta(t) = 0 \tag{2.11}$$

As constantes k_i apresentadas na Tabela 2.1 são determinadas a partir das condições iniciais e da condição de contorno, dada por (2.11). Portanto, a resposta do ângulo de carga, dada em (2.2), é função dos parâmetros K_d, K_p, K_v e ω_f.

Enquanto a solução de (2.2) é uma boa aproximação da resposta do ângulo de carga o processo de derivação por análise de pequenos sinais ignora as oscilações (*ripple*) introduzidas pelos filtros de medição de potência e ainda a variação da reatância indutiva em virtude da variação da frequência da tensão de saída do inversor.

2.2 Modelo em Espaço de Estados

Enquanto o modelo obtido através da análise de pequenos sinais consegue descrever adequadamente o comportamento geral da resposta do ângulo de carga e é especialmente

apropriado para análise da estabilidade do sistema, seu uso não é tão apropriado quando a ênfase é deslocada deste foco mais qualitativo para outro quantitativo.

Partindo do mesmo pressuposto que o inversor pode variar a sua tensão instantaneamente, o sistema apresentado na Figura 2.1 pode ser descrito por um sistema em espaço de estados. Todos os componentes do sistema podem ser considerados, sem erro apreciável, como sistemas lineares invariantes no tempo. A não-linearidade do controle decorre da multiplicação da tensão e corrente do inversor para o cálculo das potências ativas e reativas.

Portanto, para que se tenha um modelo em espaço de estados discreto basta derivar modelos discretos para a impedância de conexão e para o filtro passa-baixas. Utilizou-se a metodologia de discretização de modelos contínuos apresentada em [16], que apresenta erro nulo para pontos $t = k\tau$, para k inteiro e τ o período de amostragem.

A corrente de saída do inversor é dada por (2.12). Onde $E[k]$ e $V[k]$ são, respectivamente, os valores da tensão do inversor e da rede no instante k. A implementação desta equação em FORTRAN 95 encontra-se no Apêndice A.

$$i[k+1] = e^{-\frac{R\tau}{L}} i[k] + \frac{1 - e^{-\frac{R\tau}{L}}}{R}(E[k] - V[k]) \tag{2.12}$$

A saída y do filtro passa-baixas é dada por (2.13), sendo $u[k]$ a entrada aplicada ao filtro no instante k e ω_f a sua frequência de corte. O Apêndice B contém a implementação em FORTRAN 95.

$$y[k+1] = e^{-\omega_f \tau} y[k] + (1 - e^{-\omega_f \tau}) u[k] \tag{2.13}$$

Usando as equações (2.12) e (2.13) é possível implementar um modelo capaz de simular o sistema de controle. O código do simulador encontra-se no Apêndice C.

2.2.1 Comparação com Modelo de Pequenos Sinais

É importante comparar as repostas para ângulo de carga obtidas pelo modelo de pequenos sinais (pela resolução da equação diferencial (2.1)) com as de espaço de estados para que se possa avaliar os efeitos da linearização utilizada na obtenção do primeiro modelo.

O ponto de operação utilizado encontra-se na Tabela 2.2, na qual f representa a frequência da rede elétrica. A impedância de conexão foi $0,5 + j\,3,44\ \Omega$.

Tabela 2.2: Ponto de operação do sistema.

Variável	Valor	Unidade
δ_e	0,1558	rad
P_e	511,69	W
Q_e	80,39	var
E_e	107,11	V (RMS)
V_e	103,40	V (RMS)
f	60,00	Hz

Os parâmetros de controle para o caso sem realimentação de fase estão na Tabela 2.3 e os resultados na Figura 2.2. Observa-se que a resposta gerada por espaço de estados é mais amortecida que a de pequenos sinais.

Tabela 2.3: Parâmetros de controle para o caso sem realimentação de fase.

Variável	Valor	Unidade
K_d	0,00	rad/W
K_p	0,01	rad/J
K_v	0,01	V/var
ω_f	7,54	rad/s

O *caso base*[1] com realimentação de fase empregou os parâmetros da Tabela 2.4 e o resultado está na Figura 2.3. Como no caso anterior, nota-se um maior amortecimento no modelo por espaço de estados. Adicionalmente, constata-se que espaço de estados consegue capturar o efeito da oscilação na banda de passagem dos filtros passa-baixas.

[1] Foi escolhido como base por ter sido utilizado no trabalho que originalmente propôs a introdução da realimentação de fase: [15].

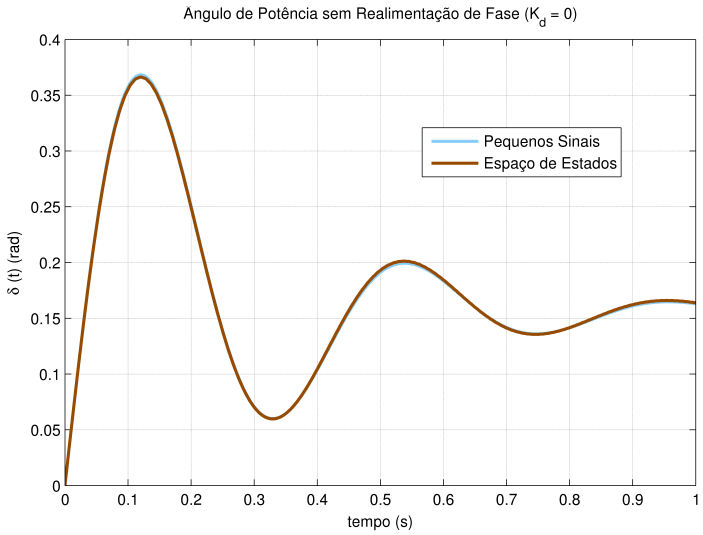

Figura 2.2: Ângulo de Potência: Modelo de Pequenos Sinais e de Espaço de Estados. $K_d = 0$.

Esta última característica é especialmente importante para que se possa avaliar o comportamento do sistema de controle para diferentes frequências de corte dos filtros.

Tabela 2.4: Parâmetros de controle com realimentação de fase. Caso base.

Variável	Valor	Unidade
K_d	0,001	rad/W
K_p	0,010	rad/J
K_v	0,010	V/var
ω_f	7,540	rad/s

Efeitos do Aumento do Ângulo de Potência

O modelo de Pequenos Sinais inclui na sua derivação a linearização das funções trigonométricas, para pequenos valores de $\Delta\theta$, adotam-se as aproximações $\text{sen}\,\Delta\theta \approx \Delta\theta$ e $\cos\Delta\theta \approx 0$. Assim sendo, espera-se que o referido modelo apresente maior divergência quanto ao comportamento real quanto maiores forem os valores do ângulo.

Com intuito de avaliar os efeitos destas aproximações, aumentou-se o ângulo de potência de $8,93^\circ$ para $20,00^\circ$. Não existem justificativas para incrementos maiores em vista de questões de estabilidade do sistema [3, 15]. Com esta mudança, o ponto de operação dado pela Tabela 2.2 passa a ser: $\delta_e = 0,3491$ rad, $P_e = 1122,44$ W, $Q_e = 146,53$ var.

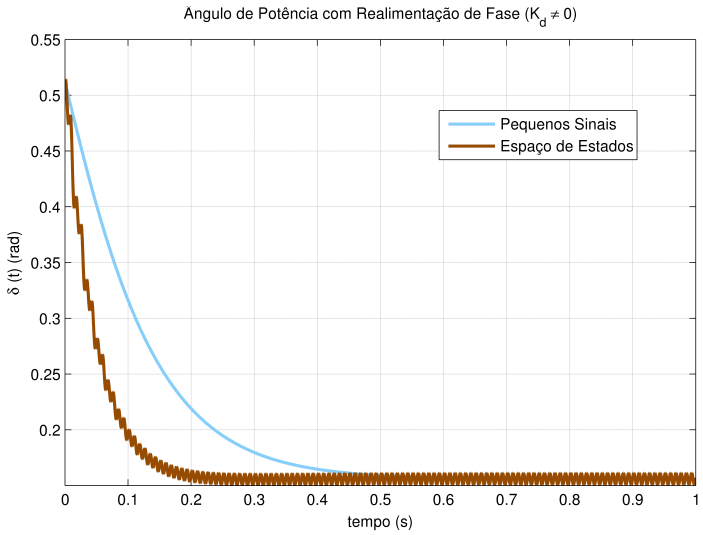

Figura 2.3: Ângulo de Potência: Modelo de Pequenos Sinais e de Espaço de Estados. Caso base.

Utilizando este novo ponto operacional e os parâmetros de controle apresentados na Tabela 2.3 foram obtidas as respostas mostradas na Figura 2.4. Diferentemente do que foi observado anteriormente, neste caso a resposta por pequenos sinais é ligeiramente mais amortecida que a de espaço de estados.

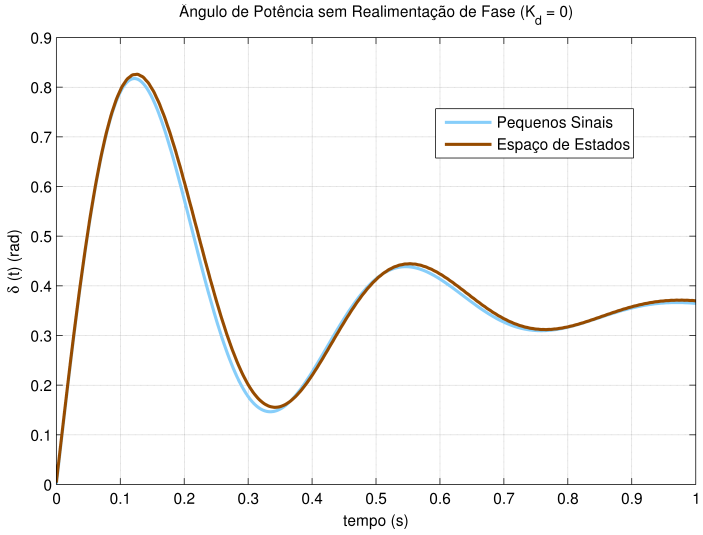

Figura 2.4: Ângulo de Potência: Modelo de Pequenos Sinais e de Espaço de Estados. $\delta_e = 20^\circ$.

A Figura 2.5 apresenta a resposta obtida para o presente ponto operacional e parâmetros de controle listados na Tabela 2.4. As características gerais das respostas são bastante semelhantes às da Figura 2.3. A diferença que mais se destaca é o aumento na oscilação da

resposta segundo espaço de estados.

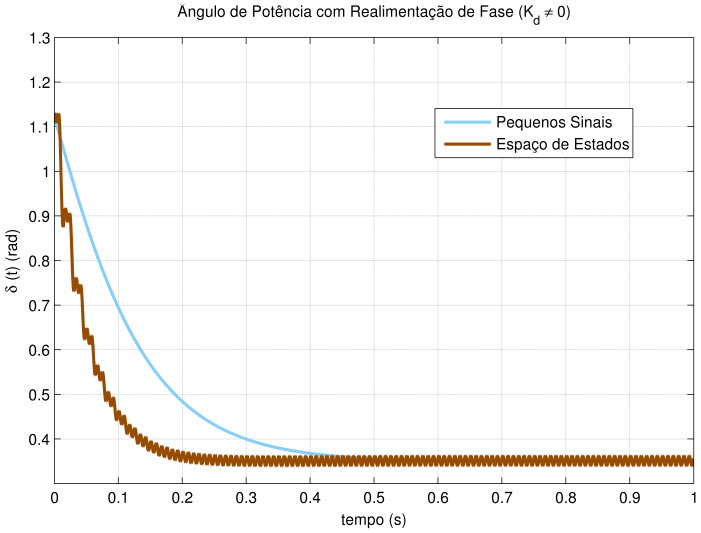

Figura 2.5: Ângulo de Potência: Modelo de Pequenos Sinais e de Espaço de Estados. $\delta_e = 20^\circ$.

Assim sendo, os efeitos da linearização não são unidirecionais para o caso sem realimentação, ou seja, o modelo de pequenos sinais não apresenta uma tendência fixa de sobre ou subestimação da resposta. No caso com realimentação, observou-se que não houve mudança apreciável em virtude da variação do ângulo: a resposta por espaço de estados é mais amortecida que por pequenos sinais e a magnitude do erro de estimação entre os modelos permanece praticamente constante.

2.2.2 Comparação com Resultados de Simulação via PSIM

Enquanto a comparação realizada na subseção anterior é instrutiva, ela não é propriamente adequada. Os méritos do modelo de espaço de estados são melhor avaliados quando ele é comparado com resultados obtidos através do *software* de simulação de circuitos PSIM.

Os resultados via PSIM foram obtidos por COELHO[2][17] e o ponto de operação empregado encontra-se na Tabela 2.5. A impedância de conexão foi mantida: $0,5 + j\,3,44\ \Omega$.

O caso base com realimentação de fase apresenta os parâmetros de controle listados na

[2]Durante a fase preparatória da publicação.

Tabela 2.5: Ponto de operação para simulação com PSIM.

Variável	Valor	Unidade
δ_e	0,1533	rad
P_e	500,00	W
Q_e	72,87	var
E_e	106,93	V (RMS)
V_e	103,40	V (RMS)
f	60,00	Hz

Tabela 2.4. Na Figura 2.6 são mostradas as respostas do ângulo de carga do PSIM e de espaço de estados. Observa-se que as respostas são bastante semelhantes quanto ao comportamento dinâmico. O Erro Absoluto Médio (EAM) considerando todos os pontos simulados foi de 0,003 radianos.

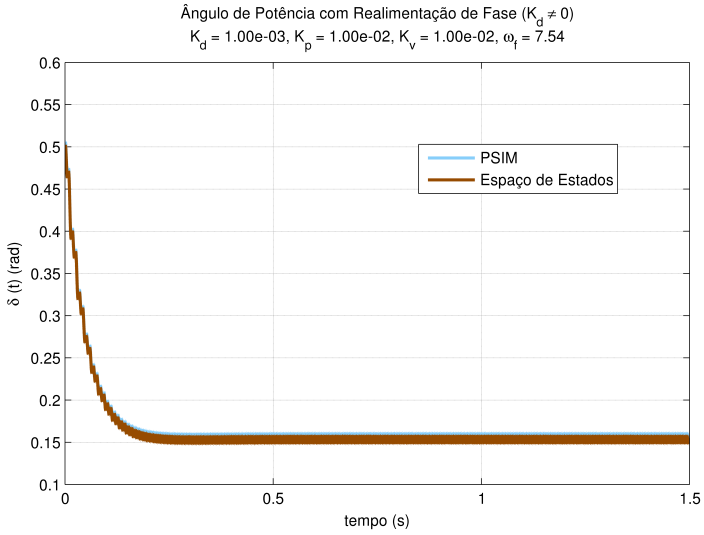

Figura 2.6: Simulação do Ângulo de Potência: PSIM e Espaço de Estados. Caso base.

A resposta do fluxo de potência ativa para o caso base pode ser vista na Figura 2.7. O EAM apurado foi de 0,13 W.

Os parâmetros de controle utilizados para definir o caso A estão nos subtítulos das Figuras 2.8 e 2.9. Os EAM's apurados foram, respectivamente, 0,003 rad e 0,67 W.

Os resultados para o ângulo de carga e fluxo de potência ativa para o caso B acham-se, respectivamente, nas Figuras 2.10 e 2.11. O EAM's associados são, respectivamente, 0,003 rad e 0,51 W.

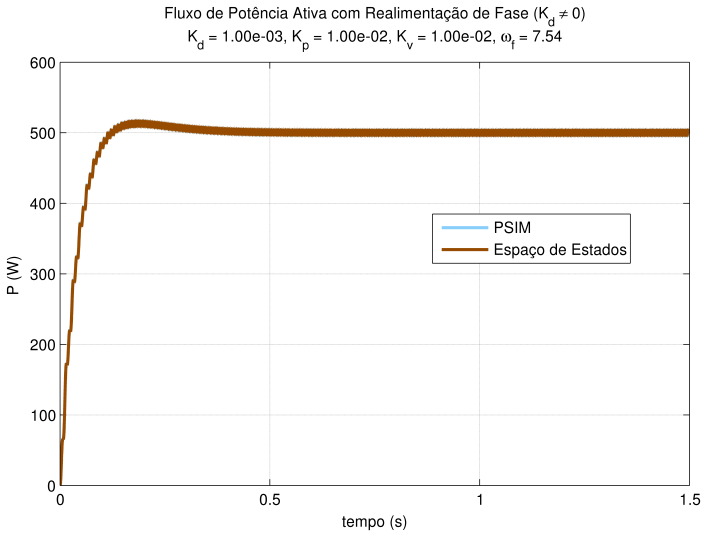

Figura 2.7: Simulação da Potência Ativa: PSIM e Espaço de Estados. Caso base.

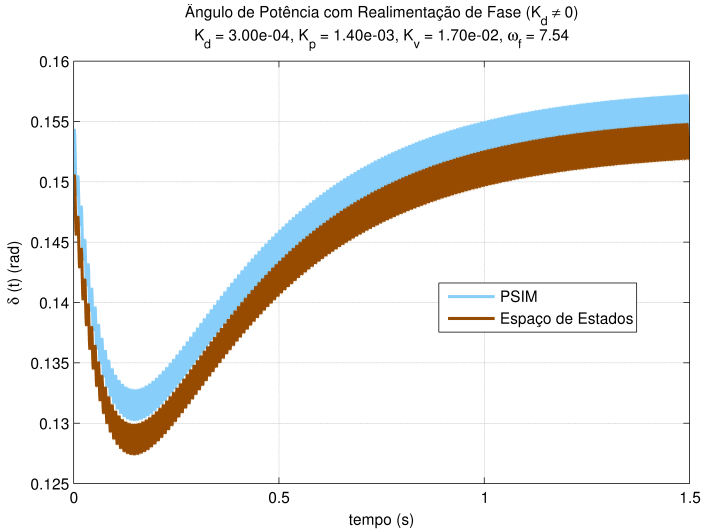

Figura 2.8: Simulação do Ângulo de Potência: PSIM e Espaço de Estados. Caso A.

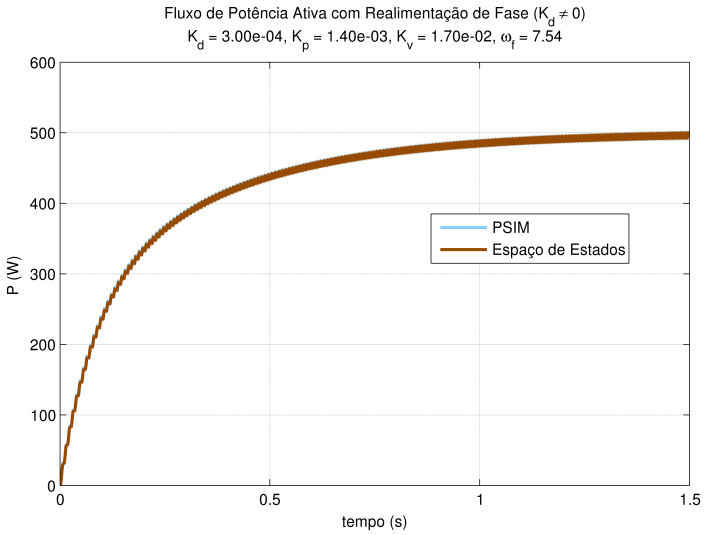

Figura 2.9: Simulação da Potência Ativa: PSIM e Espaço de Estados. Caso A.

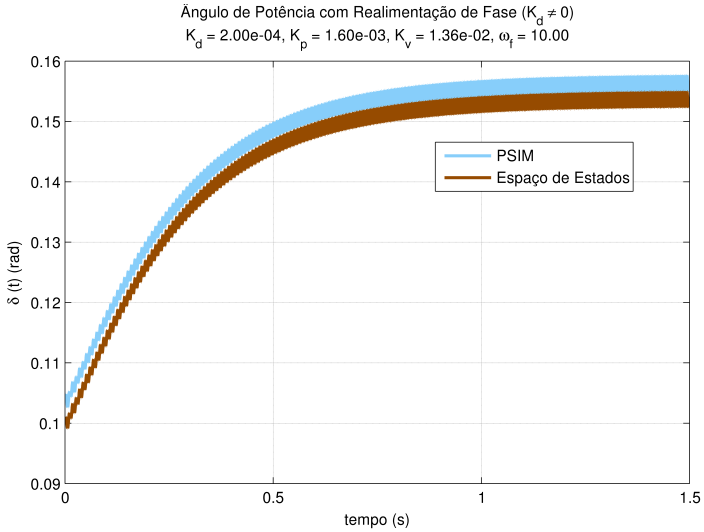

Figura 2.10: Simulação do Ângulo de Potência: PSIM e Espaço de Estados. Caso B.

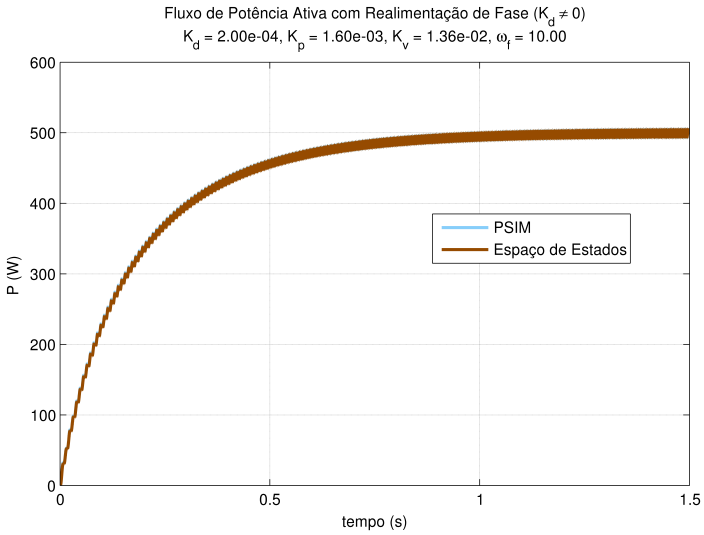

Figura 2.11: Simulação da Potência Ativa: PSIM e Espaço de Estados. Caso B.

3 Teoria de Otimização

Teoria de Otimização engloba diversas áreas da Matemática e Ciência da Computação. Ela inclui, entre outros, resultados e técnicas oriundos do Cálculo de Variações, Teoria de Controle e Programação Linear [18].

Dada a variedade de métodos e a abundância de resultados em cada subárea, este capítulo se limitará a apresentação do problema básico e uma classificação abrangente das abordagens existentes.

3.1 Introdução

O problema de otimização básico consiste em dada uma função $f : D \to \mathbb{R}$ definida no domínio $D \subseteq \mathbb{R}^n$ em determinar $\vec{x_0} \in D$ tal que:

$$f(\vec{x_0}) \leq f(\vec{x}),\ \forall \vec{x} \in D.$$

O seu dual é o problema de maximização:

$$\vec{x_0^*} \in D \quad | \quad f(\vec{x_0^*}) \geq f(\vec{x}),\ \forall \vec{x} \in D.$$

Nesta forma, o problema é equivalente ao objeto de estudo de Análise de Funções Reais. Quando são introduzidas restrições sobre o espaço de busca D tem-se o problema clássico de otimização:

$$\vec{x_0} \in S \quad | \quad f(\vec{x_0}) \leq f(\vec{x}),\ \forall \vec{x} \in S \qquad \text{, onde}$$
$$S = \{\vec{x} \quad | \quad g_i(\vec{x}) \leq 0,\ \forall \vec{x} \in D\} \cap$$
$$\{\vec{x} \quad | \quad h_j(\vec{x}) = 0,\ \forall \vec{x} \in D\}.$$

A função f é designada *função objetivo*, as funções g_i são restrições de desigualdade (*inequality constraints*) e h_i são denominadas restrições de igualdade (*equality constraints*).

A forma convencional de apresentação é como um problema de minimização. O problema de maximização é obtido negativando a função objetivo (tomando a sua inversa aditiva).

O problema é genericamente especificado da seguinte forma [19]:

$$\begin{aligned} \text{minimizar} \quad & f(\vec{x}) \\ \text{sujeita a} \quad & r_i(\vec{x}) \leq a_i, \quad i = 1, \ldots, m. \end{aligned}$$

Sendo $r_i : D \to \mathbb{R}$ as funções restrição e a_i constantes chamadas fronteiras de restrição.

3.1.1 Funções Multiobjetivo

Uma função multiobjetivo é uma função vetorial composta por n funções objetivo escalares $f_i : X \to \mathbb{R}$, $X \subseteq \mathbb{R}^m$. O problema da sua otimização pode ser descrito como [7]:

$$\begin{aligned} \text{minimizar} \quad & \vec{y} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x})) \\ \text{sujeita a} \quad & \vec{x} = (x_1, x_2, \ldots, x_m) \in X \\ & \vec{y} = (y_1, y_2, \ldots, y_n) \in Y \end{aligned}$$

Sendo $Y \subseteq \mathbb{R}^n$; $\vec{x}$ e $\vec{y}$, respectivamente, o vetor decisão e o vetor objetivo. Naturalmente, X é o espaço dos parâmetros e Y dos objetivos.

Funções multiobjetivo geralmente apresentam objetivos conflitantes. A tentativa de otimizar um dos objetivos leva à piora de outro(s). A sua importância deriva da sua aplicabilidade a projetos e processos de tomada de decisão, nos quais é necessário satisfazer múltiplos requerimentos conflitantes.

Nestes cenários não existem soluções que satisfaçam todos os objetivos simultaneamente. No projeto de CPU's, por exemplo, é desejável minimizar tanto o tempo necessário para processar uma instrução quanto o gasto de energia. Todas as maneiras práticas de minimizar o tempo de processamento (aumentar o número de transistores e/ou frequência de *clock*) aumentam o consumo de energia elétrica. Por sua vez, o processador com o menor consumo de energia possível, mantendo demais características da arquitetura, teria de ter a frequência de *clock* severamente reduzida.

Uma solução é dita ser ótima segundo o critério de Pareto quando qualquer tentativa de melhorar uma das funções objetivo resulta na piora de uma outra. Via de regra, existem múltiplos ótimos de Pareto, cada um representando um diferente compromisso entre os múltiplos objetivos. A superfície formada pela soluções ótimas é chamada *fronteira de Pareto* ou *frente de Pareto*.

Formalmente, uma solução é ótima à Pareto se, e somente se, ela não for *dominada* por

nenhuma outra solução. A relação de dominância é expressa por (3.1) [7], na qual $\vec{a} \succ \vec{b}$ lê-se: *$\vec{a}$ domina $\vec{b}$*.

$$\vec{a} \succ \vec{b} \iff f_i(\vec{a}) \geq f_i(\vec{b}),\ \forall i \in \{1,2,\ \ldots,n\} \quad \wedge \tag{3.1}$$

$$\exists j \in \{1,2,\ \ldots,n\} \mid f_j(\vec{a}) > f_j(\vec{b}) \tag{3.2}$$

A maneira mais simples de se abordar a otimização de uma função multiobjetivo é transformá-la numa função escalar pela ponderação das funções objetivo escalares que a compõe, equação (3.3). Esta é a maneira como o problema era tratado antes do desenvolvimento de meta-heurísticas multiobjetivo [12, 7].

$$y = v(\vec{x}) = \sum_{i=1}^{n} w_i f_i(\vec{x}) \tag{3.3}$$

3.1.2 Metodologias de Solução

As filosofias de solução do problema de otimização podem ser subdivididas em duas categorias: Técnicas com base Analítica e Meta-heurísticas.

As técnicas com base analítica são aquelas que exploram propriedades analíticas das funções objetivo e restrições para garantir a existência de soluções e obtê-las usando recursos computacionais eficientemente. Normalmente, os algoritmos associados a estas técnicas podem ser provados ótimos, ou seja, não existe nenhum outro algoritmo capaz de resolver a mesma classe de problema em um tempo computacional menor.

A existência de algoritmos ótimos não exclui a utilização de algoritmos de aproximação numérica. Principalmente, para os casos em que o tempo computacional demandado para resolver o problema exatamente é inviável.

No entanto, mesmo estes algoritmos de aproximação numérica são dependentes da propriedade analítica subjacente e são dotados de garantias quanto ao máximo erro de aproximação.

As meta-heurísticas não são limitadas a apenas uma classe de funções, portanto, podem lidar com problemas que não são estrita ou praticamente expressíveis em uma forma adequada à aplicação de técnicas com base analítica.

A generalidade destas metodologias implica que, quando existem, as garantias de convergência são muito mais fracas que as disponíveis nas técnicas analíticas. Em contrapartida, elas são mais facilmente aplicáveis.

Além disso, para problemas não-lineares e não-convexos não existem algoritmos gerais de solução [19]. A única outra alternativa é utilizar uma heurística especializada. Contudo, para isso, é necessário o conhecimento prévio da aplicabilidade da referida heurística e a perda de generalidade ser justificável por incremento da eficiência computacional ou precisão.

3.2 Técnicas com Base Analítica

Quando as funções objetivo e restrição são lineares o problema de otimização é chamado um programa linear (*linear program*). Caso isto não se verifique, o problema é classificado como programa não linear.

O problema é dito um *problema convexo de otimização* quando as funções objetivo e restrição são convexas. Isto significa que elas satisfazem a inequação (3.4) [19].

$$u(t\vec{x}+(1-t)\vec{y}) \leq tu(\vec{x})+(1-t)u(\vec{y}), \quad \forall \vec{x},\vec{y} \in D \qquad (t \in [0,1]) \tag{3.4}$$

A convexidade é uma condição mais genérica que linearidade. Assim sendo, otimização

convexa é uma generalização de programação linear. Outros casos particulares de otimização convexa são o problema dos mínimos quadrados e programação geométrica [19].

Segundo [19], uma grande dificuldade da aplicação de otimização convexa é reconhecer problemas que são convexos ou que podem ser reformulados como tais.

Uma vez formulados, os problemas são garantidos pela teoria de análise convexa de serem solúveis. Quanto aos algoritmos disponíveis para obter soluções, [19] propõe Mínimos Quadrados, método de Newton e Método do Ponto-Interior, em ordem crescente de generalidade e complexidade.

As garantias da análise convexa não podem ser aplicadas para problemas não-lineares que não sejam convexos. Não existem métodos de solução geral e nem mesmo garantia de existência de soluções [19].

3.3 Meta-heurísticas

As meta-heurísticas são aplicáveis não somente aos problemas tratáveis com técnicas analíticas, mas também podem lidar com funções multiobjetivo de um modo mais direto [12].

A revisão bibliográfica conduzida por [12] abrangeu 115 artigos referentes a meta-heurísticas aplicadas a programação multiobjetivo no período de 1991–1999. Constatou-se que 70% dos artigos utilizaram algoritmos genéticos como técnica meta-heurística primária, 24% usaram recozimento simulado (*simulated annealing*) e 6% busca tabu. Aplicações de colônias de formigas (para funções multiobjetivo) não foram encontradas.

É conveniente classificar as meta-heurísticas de acordo com a sua inspiração filosófica em evolucionárias e não evolucionárias. As evolucionárias empregam diretamente mecanismos que simulam o processo biológico de seleção natural.

Ambas abordagens apresentam componentes aleatórios na exploração do espaço de busca. Elas diferem de um busca completamente aleatória por empregarem algum princípio que orienta a exploração em determinada direção. Muitas vezes o grau de aleatoriedade é controlável por um dos parâmetros da meta-heurística e para valores extremos deste parâmetro a heurística se degenera numa busca inteiramente aleatória.

A intensidade da influência do princípio que guia a exploração geralmente é um dos parâmetros da meta-heurística. Em decorrência da generalidade destes algoritmos, a interação entre este parâmetro e os demais é necessariamente complexa e tem um impacto significativo na efetividade do uso destas técnicas.

A sintonia dos referidos parâmetros deve ser feita experimentalmente para o problema que se deseja resolver. O ponto de partida são as recomendações gerais para a técnica que se está empregando ou, quando disponível, recomendações mais específicas para o tipo do problema.

Em virtude disto, existe extensiva literatura propondo modificações às meta-heurísticas para aproximá-las o mais possível de autosintonizáveis.

3.3.1 Não Evolucionárias

Entre as meta-heurísticas não baseadas em aspectos evolutivos estão busca por padrão (*pattern search*) [20, 8], recozimento simulado, busca tabu, métodos de Monte Carlo e método da entropia cruzada (*cross-entropy method*) [21]. Destas serão brevemente revisadas apenas recozimento simulado e busca tabu por serem as metodologias mais aplicadas [12].

Técnicas de desenvolvimento mais recente, como Otimização por Enxame de Partículas (*Particle Swarm Optimization*) [22, 23] e Evolução Diferencial (*Differential Evolution*) [24], não serão consideradas.

Recozimento Simulado

O recozimento simulado se baseia nas mudanças na estrutura cristalina de metais durante o processo de recozimento utilizado em metalurgia.

O metal ao ser aquecido tem seus átomos promovidos a estados mais energéticos, liberando aqueles que estavam em estruturas cristalinas indesejadas. Durante o resfriamento controlado os átomos se rearranjam em configurações de forma a minimizar o estado energético. Isto implica em configurações cristalinas mais estáveis, ou seja, menor número de defeitos.

Os estados iniciais indesejados são análogos a mínimos locais e o resfriamento controlado é feito para aumentar a probabilidade que o mínimo global será obtido. Como as demais meta-heurísticas, não existe garantia que ótimo global será atingido, apenas que ele será aproximado.

A função objetivo do recozimento simulado é análoga à energia interna no caso do metal. O algoritmo é estruturado de forma a minimizá-la.

O processo se inicia com uma alta temperatura e um ponto aleatório do espaço de busca. A cada passo se decide probabilisticamente se a solução candidata atual será substituída por uma nova.

A distribuição de probabilidade é uma função tanto da temperatura como da diferença de energia entre os estados. Quando a temperatura é alta, a probabilidade de aceitação de estados piores também é alta. Conforme a temperatura vai diminuindo, a aceitação de soluções candidatas mais energéticas diminui; tornando-se praticamente nula na etapa final do algoritmo quando a temperatura se aproxima de zero.

No contexto de otimização com múltiplos objetivos, [13] compara duas versões de recozimento simulado com algoritmos evolucionários. Todos os algoritmos comparados apre-

sentam adaptações específicas para o referido contexto.

Os algoritmos evolucionários avaliados foram PAES e NSGA-II. O primeiro, *Pareto Archived Evolution Strategy*, é uma estratégia evolutiva com população de único indivíduo, onde seleção atua sobre a população intermediária formada pelo pai e pelo filho. A codificação utiliza uma sequência binária e existe um "arquivo" (*archive*) onde as melhores soluções, segundo o critério de Pareto, são mantidas.

O segundo é um algoritmo genético elitista com função de adequação que mede o quanto uma solução não é dominada pelas demais soluções presentes na população. O método de seleção empregado é o torneio binário e o algoritmo é avaliado tanto para representações por sequências binárias como por ponto flutuante.

Foram consideradas duas versões de recozimento: MOSA (*Multiobjective Simulated Annealing*) e AMOSA (*Archived Multiobjective Simulated Annealing*). A primeira utiliza uma função energia que incorpora uma estimativa do grau de dominância e já havia sido proposta anteriormente.

A última versão é introduzida em [13] e emprega uma função energia distinta. Além disso, ela utiliza um arquivo para armazenamento de soluções não-dominadas cujo tamanho máximo deve ser definido pelo usuário. Clusterização é empregada quando o número de soluções ultrapassa o tamanho do arquivo. Ela é realizada de maneira a garantir a diversidade das soluções.

Nos testes comparativos realizados constatou-se que PAES teve o pior desempenho e que AMOSA é melhor que MOSA e NSGA-II na maioria dos casos. Adicionalmente, as soluções providas por AMOSA foram mais diversificadas que as do NSGA-II em todos os problemas avaliados. Contudo, os próprios autores observam que as modificações responsáveis pelo desempenho superior de AMOSA podem ser incorporadas em outros algoritmos [13].

É interessante destacar, ainda, que AMOSA é comparativamente mais distante da versão

básica de recozimento que o algoritmo genético considerado é do algoritmo genético básico.

Busca Tabu

A busca tabu em sua forma mais simples explora a vizinhança de uma solução não aceitando soluções candidatas que já foram avaliadas no passado recente.

A lista contendo as soluções já exploradas é designada *lista tabu* e apresenta um tamanho limitado. Por isto, uma solução proibida (tabu) pode voltar a ser aceita com o transcorrer do algoritmo.

A geração de novas soluções a partir da solução corrente pode empregar técnicas variadas. No entanto, a busca tabu é essencialmente uma busca local [14].

As características da busca tabu podem ser alteradas de acordo com o uso da memória. Assim sendo, ao usar uma memória com prazo de expiração longo o algoritmo pode detectar casos em que a busca ficou presa a um ótimo local e mover a procura para um nova região.

Esta heurística é mais adequada para ser aplicada a casos discretos, especialmente problemas de otimização combinatorial. Ela pode ser adaptada para otimização utilizando variáveis em ponto flutuante mudando a metodologia de rejeição.

Ao invés de rejeitar uma solução que é exatamente igual a outra já presente na lista tabu, rejeitam-se todas as soluções que estejam dentro de certa distância das soluções tabu. Isto evita que a lista tabu se torne completamente redundante (com o critério de igualdade estrita o algoritmo aceitaria todas as soluções arbitrariamente próximas daquelas na lista tabu).

Dois algoritmos de busca tabu para otimização de funções multiobjetivos são comparados com NSGA-II em [14]. Ambas buscas tabu utilizam estruturas de memória de curto, médio e longo prazo.

A memória de curto prazo é empregada para manter a lista de soluções tabu propriamente dita, enquanto a de médio prazo é usada para direcionar a busca para áreas promissoras do

espaço de busca. Por sua vez, a memória de longa duração serve para manter a diversidade das soluções, direcionando a busca para regiões pouco exploradas.

Ambos algoritmos tabu codificam as variáveis como números reais. No entanto, diferem quanto a metodologia empregada para realizar a etapa de busca local.

As buscas tabu e o algoritmo genético foram avaliados para seis funções multiobjetivo sintéticas da família ZDT. Cada algoritmo foi executado 45 vezes para 1.000, 5.000 e 10.000 avaliações da função objetivo.

Considerando apenas as variantes tabu, a busca introduzida no artigo foi melhor em 60% dos casos que a variante previamente disponível.

O algoritmo genético NSGA-II foi melhor que ambas em 47% dos cenários. Ele também foi pior que elas em 43% dos cenários. A variante tabu introduzida é considerada competitiva com NSGA-II, especialmente para problemas com muitas restrições [14].

É importante mencionar que [14] destaca que só existia uma variante prévia de busca tabu aplicada a otimização de funções multiobjetivos com valores reais [25]. Além disso, os autores notam que a primeira vertente de algoritmo genético para funções multiobjetivo foi desenvolvida em 1985.

3.3.2 Algoritmos Evolucionários

Algoritmos Evolucionários (AE's) empregam simulações do processo básico de seleção natural. Uma de suas características distintivas é a presença de uma população de soluções candidatas. Num determinado instante existem vários indivíduos distintos representando pontos do espaço de busca com diferentes valores da função objetivo.

Eles variam quanto à forma em que o espaço de busca é representado; a dinâmica de emprego dos operadores genéticos, a forma como eles são parametrizados e expostos. Historicamente, algoritmos genéticos (AG's) e estratégias de evolução (ES — *evolution strategies*)

são as formas básicas de algoritmos evolucionários [6].

Os AG's enfatizam o uso de recombinação sobre o uso de mutação, enquanto nas ES o enfoque é o reverso. A principal área de aplicação de ES é otimização [6], enquanto AG's são tradicionalmente considerados multipropósito [6, 26, 27].

Existem duas vertentes de AE's voltados para aplicações computacionais: programação genética e programação evolucionária [6]. A primeira representa computações como árvores sintáticas abstratas (*Abstract Syntax Trees*), enquanto a segunda trabalha com máquinas de estado. Ambas são voltadas para programação automática e aprendizado de máquina, não sendo boas candidatas para otimização.

ES frequentemente utilizam representação em ponto flutuante para as variáveis a serem otimizadas e não existe a noção de processamento de esquemas (*schema*) [6]. ES foram pioneiras em utilizar diferentes métodos de manipulação das populações de pais e descendentes. Adicionalmente, elas exploraram mecanismos de autoadaptação aos espaços de buscas comparativamente cedo [6].

Normalmente, ES emprega taxas de mutação que seguem uma distribuição normal ou normal-logarítmica e os parâmetros da distribuição são codificados juntos com as variáveis a serem otimizadas. Contudo, mesmo quando se empregam técnicas autoadaptativas (que exploram propriedades estatísticas do espaço de busca) para gerar novos indivíduos, ainda é preciso determinar o valor de certos parâmetros [6].

ES apresenta resultados melhores quando comparada com o AG básico. No entanto, quando o AG é modificado para ter manipulações populacionais mais diretamente comparáveis às utilizadas por ES, o AG torna-se mais competitivo [6].

As iterações mais recentes de AG's procuram contornar deficiências observadas no algoritmo genético básico ou adaptá-lo para um novo domínio de aplicação.

No caso de otimização de funções multiobjetivos, [7] compara 4 variantes de algoritmos

evolucionários na solução de um problema teórico (*knapsack* 0/1 multiobjetivo) e um caso prático (síntese a nível de sistema — *system-level synthesis*). Constata-se que quanto mais intensamente adaptados ao domínio, melhores os resultados. Em contrapartida, há perca de generalidade e aumento da complexidade computacional.

Cabe ainda ressaltar os motivos que levaram [7] a não considerar a implementação de um algoritmo genético elitista simples: o fato de convergirem para uma única solução. No contexto do referido artigo, é desejável que a população ao final do algoritmo contenha o maior número de soluções ótimas de Pareto distintas.

4 Algoritmos Genéticos

4.1 Funcionamento Básico

Algoritmo genético (AG) é uma técnica de busca global baseada no processo de evolução biológico que exibe auto-organização e adaptação como consequências unicamente de sua experiência com o ambiente [27]. O fluxograma de um AG básico encontra-se à Figura 4.1.

O AG utiliza uma população de indivíduos em cada iteração, que é chamada geração. Cada indivíduo, ou cromossomo, representa uma possível solução para o problema a ser resolvido. O ambiente onde se processa a evolução é representado por uma função que associa a cada cromossomo um valor que indica o seu grau de adequação (*fitness*). Portanto, a função adequação deve expressar a(s) característica(s) que se deseja(m) otimizar e o cromossomo deve representar os parâmetros disponíveis.

Uma vantagem dos AG's é que a função adequação não necessita ser definida para todos os possíveis cromossomos, nem precisa exibir propriedades como continuidade e diferenciabilidade.

O AG inicia-se com uma população gerada normalmente de maneira aleatória. Cada indivíduo tem o seu valor de adequação determinado. Os cromossomos com maior valor de adequação apresentam maior chance de serem selecionados para reprodução, ou cruzamento (*crossover*). Deste modo, indivíduos mais aptos tem maior probabilidade de contribuir para a formação da nova geração. Consequentemente, a tendência é que a combinação das carac-

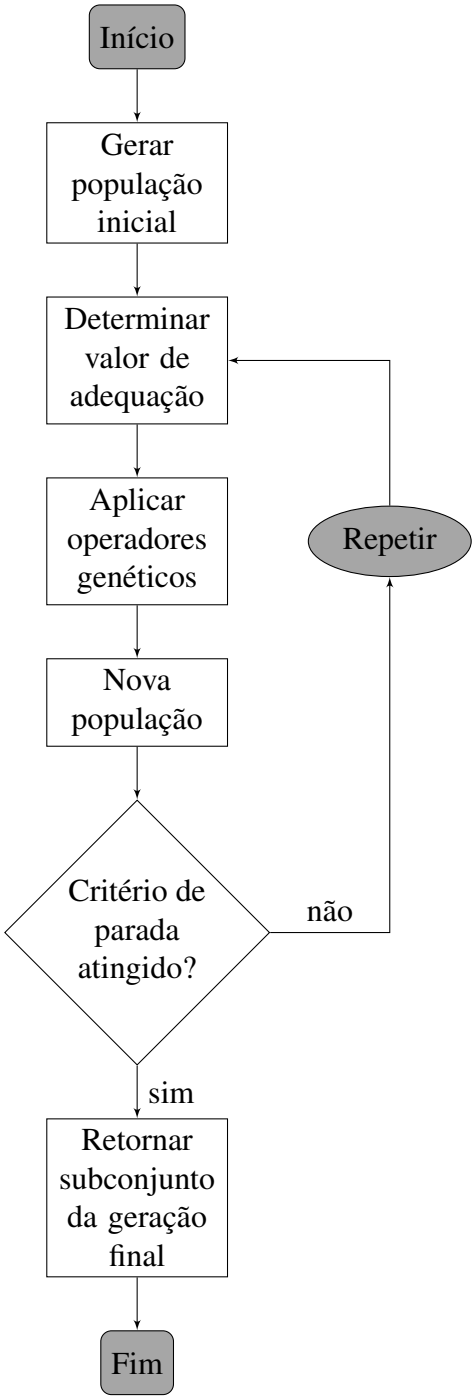

Figura 4.1: Fluxograma de um Algoritmo Genético Simplificado.

terísticas de cromossomos aptos resultem em indivíduos ainda mais adequados.

A reprodução, portanto, é uma maneira mais genérica e robusta de explorar a vizinhança de regiões promissoras do espaço de busca. É importante ressaltar que essa "vizinhança" é relativa a forma como o cromossomo é codificado e está sujeita aos efeitos da aleatoriedade do(s) ponto(s) de cruzamento.

Após o cruzamento aplica-se a mutação. O processo de mutação independe do valor de adequação do indivíduo e ocorre aleatoriamente com uma determinada probabilidade, a taxa de mutação. O principal objetivo do processo de mutação é a exploração de novas aéreas do espaço do busca, sendo o principal mecanismo pelo qual o AG evita ficar restrito a ótimos locais.

Concluída a etapa de mutação tem-se a nova geração. Por sua vez, esta população será novamente sujeita aos operadores genéticos reprodução e mutação para formar a geração seguinte. O processo se repete até que um critério de parada adequado seja atingido.

Os critérios de parada mais utilizados são um número máximo de gerações ou uma medida de convergência da população em torno de uma solução. Uma maneira comum de avaliar a convergência é definir um número máximo de gerações consecutivas em que o melhor indivíduo não tem o seu valor de adequação incrementado.

4.2 Evolução Biológica

Uma vez que a metodologia do AG é inspirada pelo processo natural de evolução biológica é importante ressaltar algumas características desta última.

A principal delas é que o processo de seleção natural não é, *qualitativamente*, um processo de otimização[1]. Para que este fosse o caso, seria necessário que o processo de seleção natural, ou *Natureza*, possuísse agência e intencionalidade.

[1]Tecnicamente, ele o é. Até o ponto em que o processo de seleção possa ser considerado isomórfico a um

Justamente o grande desafio da Teoria da Evolução é descrever os mecanismos pelos quais é possível que um sistema desprovido de inteligência e propósito atue de maneira a modificar organismos tornando-os cada vez mais adaptados ao seu ambiente.

De fato, por vezes, tão bem adaptados que *aparentemente* seriam necessários um propósito, ou mesmo um projeto, mas até presciência.

A grande virtude da Teoria Evolutiva é a simplicidade com que estes processos são explicados. Sem contar que ela é de fato uma teoria científica[2]. Além disso, ela resistiu tentativas de refutação ou alterações desde 1859[3]e contínua sendo a teoria científica mais adequada para explicar os processos biológicos [30].

A essência do processo evolutivo pode ser caracterizada como a seleção dos indivíduos mais aptos. Sendo importante entender como se dá a seleção e o que é aptidão.

O processo de seleção é totalmente passivo. Em um dado momento, ele só ocorre sobre indivíduos já existentes e com características previamente definidas.

A seleção ocorre no processo reprodutivo do organismo; tanto para reprodução sexuada como assexuada. É importante notar que do ponto de vista evolutivo toda a extensão da vida do organismo é parte do processo reprodutivo.

Deste modo, se entende que organismos com defeitos congênitos *graves* sofrem uma *pressão seletiva* [26] que, com o passar das gerações, tem o efeito de eliminá-los do repositório genético da espécie.

O mecanismo específico em que a seleção ocorre, neste caso, é que estes indivíduos são simplesmente muito debilitados e perecem antes de se reproduzir. Assim sendo, seu código genético não passa para a geração seguinte.

algoritmo de busca.

[2]Seguindo a definição proposta por Karl Popper [28], em especial, a exigência que para ser considerada científica uma teoria deve ser passível de refutação experimental [29].

[3]*On the Origin of Species*, de autoria de Charles Darwin, foi publicado em 24 de novembro de 1859.

Note-se que a gravidade — e mesmo a caracterização como defeito — da condição genética só pode ser feita depois que se verifica o insucesso reprodutivo do organismo. E, na grande parte dos casos, este insucesso é relativo e deve considerar a taxa de reprodução típica da espécie à qual o indivíduo pertence.

Adicionalmente, a seleção só atua sobre características de uma população que influenciam significativamente o sucesso reprodutivo. As demais características, em virtude da sua baixa correlação com reprodutividade, tendem a se encontrar aleatoriamente dispersas na população. Explicando, parcialmente, a variabilidade intraespecífica.

No entanto, se uma destas características passa a ter uma correlação significativa com sucesso reprodutivo, a pressão de seleção crescente fará com que uma proporção crescente da população a tenha a cada geração subsequente.

A mudança na pressão de seleção é dada pelas mudanças do ambiente em que a espécie está inserida. É importante notar que própria população faz parte deste ambiente e, portanto, afeta o processo de seleção.

Isto é especialmente evidente nos casos de coevolução de espécies e a dinâmica das relações interespecíficas resultantes: presa/predador, vetor/parasita [31] e simbioses [26].

Ainda, é importante notar que a associação intuitiva de evolução com um resultado positivo (progresso) *não* é consistente com a evidência obtida observando o mundo natural. É suficiente lembrar que, de todas as espécies que já existiram, estima-se que 99,9% se encontram extintas [32].

Este fato, por si só, pareceria sugerir que o processo evolutivo é extremamente ineficiente. No entanto, esta intuição é problemática. Primeiro, porque só se tem conhecimento de um único planeta em que este processo ocorreu [33].

Segundo, que existem outros fatos que devem ser considerados. Uma característica fundamental da Teoria Evolutiva é que o processo de adaptação ocorre gradualmente, isto é, não

existem "saltos evolutivos". No entanto, isto não implica que os efeitos da seleção natural só podem ser observados em longas escalas temporais.

Isto se verifica especialmente na domesticação de animais [34], estudo do registro fóssil [30] e infectabilidade de patogêneos [31]. Mas, a evidência mais contundente provém da análise comparativa do material genético das espécies catalogadas que indica, com alta probabilidade, que todas as formas de vida hoje existentes descendem de um único organismo unicelular [35, 36].

4.2.1 Transcrição Genética e Características Físicas

Não só o processo em que o código genético afeta as características de um organismo é altamente não-linear, ele também não é um processo inversível. Isto é verdadeiro mesmo quando se desconsidera erros de transcrição.

A presença de um gene, por si só, não implica que a característica por ele codificada se manifestará, por quanto tempo ou em que local. Isto é singularmente claro em organismos multicelulares, nos quais todas as células possuem o mesmo material genético.

No entanto, num organismo saudável em cada região somente um subconjunto específico dos genes estarão ativos. Formando as estruturas diferenciadas designadas como tecidos.

4.3 Implementação de Algoritmos Genéticos

Um algoritmo genético basicamente deve apresentar os seguinte componentes:

- função adequação (função *fitness*);
- codificação das variáveis a serem otimizadas;
- metodologia de cruzamento;

- metodologia de mutação.

Enquanto existe uma considerável liberdade na escolha de cada um destes itens o seu grau de correlação sempre é não nulo. Esta maleabilidade deve ser utilizada para aproximar as estruturas de inter-relação originalmente presentes no problema que se deseja otimizar.

O passo inicial na aplicação de AG's é selecionar quais as variáveis que deverão ser consideradas para o processo de otimização. Em primeira instância deveriam ser selecionadas todas aquelas que contribuem significativamente para as saídas de interesse.

No entanto, deve ser considerada a complexidade e a sensibilidade da função adequação. Quanto maior a complexidade, de modo geral, maior será o poder de processamento necessário para avaliar a função *fitness*.

Deste modo, é preciso considerar que (para um mesmo tempo de processamento) o uso de uma função mais custosa deverá incorrer na redução do tamanho da população e/ou número de gerações que serão empregados. Dependendo do comportamento de convergência do problema tal alternativa é bastante razoável. Especialmente, para problemas em que a convergência se dá de maneira relativamente rápida.

Além do aspecto computacional, a escolha da função adequação modifica as características de convergência do AG. Apesar desta dependência ser altamente complexa podem se salientar algumas qualidades desejáveis.

Na impossibilidade da função ser contínua, é interessante que ela seja escolhida de modo a introduzir o mínimo de descontinuidades possíveis. Esta diretriz é relevante na escolha do mecanismo de penalidades.

Apesar de AG's poderem lidar com penalidades altamente descontínuas, o seu desempenho é prejudicado. Normalmente, o desempenho sempre será melhor quanto maior for a suavidade nas transições de *fitness* entre uma solução aceitável e os seus vizinhos na representação genética escolhida que codificam soluções rejeitadas.

Adotando esta metodologia, implica que as transições de valor de adequação entre soluções rejeitadas também não sejam abruptas.

Além destes aspectos, é recomendável optar por funções de adequação cujo espaço de busca tenham a maior quantidade de estrutura possível.

O exemplo clássico para ilustrar este ponto é o Problema do Caixeiro Viajante (TSP — *Travelling Salesman Problem*). O problema consiste em dada uma lista de cidades em determinar o roteiro com a menor distância total que visite todas as cidades uma única vez partindo e retornando para a cidade de origem.

Este problema apresenta complexidade computacional *NP-Hard* e, portanto, não possuí solução genérica que possa ser computada num intervalo de tempo finito. Logicamente, isto não inviabiliza a determinação de soluções exatas para roteiros com pequeno número de cidades.

Contudo, para muitos casos práticos na área de Pesquisa Operacional o número de cidades é maior do que é exatamente otimizável num tempo aceitável. Assim sendo, a maior parte das aplicações do TSP utilizam técnicas heurísticas, dentre as quais AG's.

Neste contexto, observa-se que para um mesmo número de cidades o AG tem mais dificuldade em encontrar roteiros ótimos quanto mais distantes elas estão entre si. Como não houve variação na complexidade computacional do problema, outro fator deve ser responsável.

Quanto maior a separação entre cidades mais evidente é o efeito da curvatura terrestre e menos euclidianas se tornam as distâncias. A desigualdade triangular se torna inválida e o espaço de busca perde uma estrutura que facilitava a sua exploração.

O TSP também é um caso de estudo interessante para AG's porque, apesar de apresentar bom desempenho, eles não estão entre as heurísticas mais eficientes. Entretanto, as melhores técnicas são específicas para a solução de TSP. O custo do aumento da efetividade é a perda

de generalidade.

4.3.1 Representação Genética/Cromossômica

De modo geral, existem duas metodologias de representação da informação genética para implementação de AG's em computadores digitais. A primeira delas consiste em codificar todas as variáveis de interesse em uma única sequência de bits de tamanho fixo. A segunda representa diretamente cada variável como um número em ponto flutuante.

Do ponto de vista de custo computacional, a determinação de qual método é mais vantajoso depende da aplicação. A tendência é que para casos com pequeno número de variáveis (possam ser representadas em 64 bits ou menos) o desempenho seja próximo. Variando com a arquitetura do processador e até mesmo com a versão de uma mesma arquitetura de um mesmo fabricante.

Conforme o número de variáveis aumenta ou necessidade de faixa dinâmica o custo computacional da implementação em ponto flutuante se torna menor.

A facilidade de implementação em ponto flutuante também é maior e exige menos cuidados iniciais. Caso sejam necessárias fazer adaptações ou incluir variáveis a codificação em ponto flutuante é mais prática.

Representação Binária de Comprimento Fixo

A codificação de comprimento fixo requer que os valores máximo e mínimo de cada variável sejam explicitamente determinados. A precisão de cada variável também precisa ser definida de antemão.

Para uma variável v_i com valor máximo $\max v_i$, mínimo $\min v_i$ e com precisão p_i o mínimo número de bits necessário para representá-la é dado pela equação (4.1).

$$b_i = \left\lceil \log_2 \left(\frac{\max v_i - \min v_i}{p_i} \right) \right\rceil \tag{4.1}$$

Uma vantagem da codificação de comprimento fixo é que uma variável sempre conservará a sua precisão. Ela é independente do número de operações às quais a variável é submetida. Além disso, o resultado de qualquer combinação de operações aritméticas a que a variável for sujeita será o mesmo. Ele é independente de permutações na ordem das operações *matematicamente* comutativas.

A presente codificação especifica um espaço de busca de tamanho definido e fixo, que pode ser caracterizado pelo número total de bits utilizados para codificar as suas variáveis.

O número mínimo de bits B necessário para representar n variáveis é dado por (4.2).

$$B = \sum_{i=1}^{n} b_i \tag{4.2}$$

Assim sendo, é possível estabelecer métricas de desempenho relacionadas ao tamanho do espaço de busca para avaliar a efetividade do AG.

A fração do espaço de busca explorado pelo AG é dada pela equação (4.3), onde $n*^{i}_{Cr}$ é o número de cromossomos únicos que o AG avaliou durante a i-ésima execução.

$$f^{i}_{explorado} = \frac{n*^{i}_{Cr}}{2^B} \tag{4.3}$$

Quanto maior a fração mais próximo o AG está de uma busca exaustiva. Deste modo, pequenos fatores são desejáveis.

Outra métrica significativa é o fator de diversidade dado por (4.4), onde n^{i}_{Cr} é o número total de cromossomos avaliados pelo AG durante a execução i. Ele permite avaliar o custo computacional do AG, orientando variações na implementação do programa.

$$f^i_{diversidade} = \frac{n*^i_{Cr}}{n^i_{Cr}} \tag{4.4}$$

Representação em Ponto Flutuante

Esta representação codifica cada variável como um número de ponto flutuante. No passado havia uma maior variedade de formatos de número flutuante. No entanto, na última década prevaleceu o padrão IEEE-754.

O *IEEE Standard for Floating-Point Arithmetic* (IEEE-754) define o formato de representação binário de números de ponto flutuante, as suas operações e características. A publicação original ocorreu em 1985 e a revisão mais recente em 2008.

Os formatos relevantes para implementação de AG's são os de precisão simples, dupla e quádrupla. Os formatos de precisão simples costumavam ser mais empregados em virtude de serem computacionalmente menos exigentes.

Nos últimos 5 anos o desempenho de números de precisão dupla tem sido aproximadamente o mesmo que os cálculos de precisão simples. Como a precisão quádrupla ainda incorre em um tempo de processamento maior, somente aplicações que realmente necessitam da precisão extra a utilizam.

Independente da precisão a representação binária destes formatos é subdividida em três partes: um bit de sinal; um significando (mantissa, na versão anterior da especificação) e expoente.

Todos os formatos anteriormente mencionados utilizam base 2. Somente na revisão de 2008 foram incluídos formatos que utilizam base 10.

Enquanto qualquer número dentro da faixa abrangida por determinado formato pode ser representado em base 2, muitos números que em base 10 tem um representação finita geram uma representação infinita em base 2. Consequentemente, ela tem de ser arredondada para

ser representável no número de bits disponíveis. Os erros introduzidos são inaceitáveis para certas aplicações, notavelmente para cálculos envolvendo valores monetários.

O formato de precisão simples propicia *aproximadamente* o equivalente a sete dígitos decimais de precisão e pode representar valores compreendidos entre $-3,4 \times 10^{38}$ e $3,4 \times 10^{38}$. O menor número representável é $1,4 \times 10^{-45}$.

A precisão dupla proporciona o equivalente a quinze dígitos decimais de precisão e pode representar valores compreendidos entre $-1,8 \times 10^{308}$ e $1,8 \times 10^{308}$. O menor número representável é $5,0 \times 10^{-324}$.

Qualquer que seja a precisão, a ordem com que as operações aritméticas são executadas resultam em valores diferentes. Ainda mais relevante é que a reorganização de operações matematicamente comutativas tem impacto significante no erro de aproximação.

Isto é especialmente relevante para algoritmos que realizam múltiplas interações de cálculos de ponto flutuante, devido ao acumulo sucessivo de erros.

Uma análise compreensiva sobre estes problemas e recomendações para implementação amparadas por análise numérica dos erros de arredondamento se encontra em [37].

Enquanto a representação em ponto flutuante não dispõe da correspondência direta entre o tamanho do espaço de busca e a codificação binária, o Teorema de Schemata continua válido [38]. Segundo [38], a representação em ponto flutuante propicia maior eficiência, maior precisão e maior liberdade na seleção de técnicas de cruzamento e mutação.

A afirmação que a representação em ponto flutuante apresenta maior precisão pode ser entendida como válida numa parcela significativa dos casos práticos. Contudo, é importante salientar quais são essas condições e retificar a definição de precisão.

A precisão de uma representação é o número de dígitos fracionários que podem ser codificados exatamente. Como exposto anteriormente, a representação binária de comprimento

fixo sempre mantém a sua precisão ao custo de possuir uma faixa menor de valores representáveis.

Já a representação em ponto flutuante possui uma precisão variável. No entanto, consegue representar uma faixa muito maior de valores utilizando o mesmo número de bits.

A máxima precisão da codificação em ponto flutuante se dá para valores no intervalo $[-1, 1]$. Conforme as ordens de magnitude aumentam, esta representação perde dígitos fracionários.

Contudo, na maioria dos problemas a precisão propiciada por números de ponto flutuante é suficiente; mas é importante ficar atento ao número de operações a que as variáveis estarão sujeitas bem como a ordem em que serão realizadas.

4.3.2 Critérios de Parada

Dado o caráter estocástico dos AG's e sua aplicação principalmente para problemas onde não é viável encontrar a solução global torna-se necessário definir critérios para quando a execução do algoritmo deve ser encerrada.

As técnicas disponíveis podem ser amplamente enquadradas em duas categorias: relacionadas ao esforço computacional e relacionadas à convergência.

As principais metodologias correlacionadas ao esforço computacional presentes na literatura são número de gerações [39, 27, 26] e tempo total de execução.

Conquanto a interferência do agendador do sistema operacional e demais programas executando no sistema seja pequena ambas as métricas são perfeitamente equivalentes. No entanto, o número de gerações ainda será uma medida confiável do esforço empregado pelo AG ainda que estas interferências sejam consideráveis.

A divergência se torna cada vez mais significativa quanto maior for o tempo médio de

execução do AG. Contudo, o tempo total de execução é mais prático quando o AG é implementado num *cluster* de computadores para o qual o uso seja delimitado pelo tempo de processamento.

Entre as técnicas relacionadas à convergência a mais simples é interromper a execução após certo número de gerações sem melhoria na função adequação, por exemplo, [40] aborta a execução do AG após 100 gerações consecutivas sem melhorias no *fitness* (quando não comparando AG's com programação linear inteira).

Outra filosofia consiste em estabelecer uma métrica de convergência adequada ao problema. Por exemplo, [39] usa uma função distância entre soluções para definir um índice de convergência. Enquanto este índice for crescente o AG continua. Adicionalmente, a cada G gerações o índice é avaliado e, caso ele tenha atingido um limiar prefixado, o algoritmo é interrompido.

4.3.3 Convergência de Algoritmos Genéticos

O Algoritmo Genético Clássico (AGC) não converge para o ótimo global, mesmo tendo um tempo infinito de computação disponível [41]. Este resultado é independente das influências da população inicial, dos valores das taxas de cruzamento e mutação.

Para que o AGC convirja, conforme demonstrado em [41], para o ótimo global é necessário que as taxas de mutação e cruzamento variem durante a execução do algoritmo. Outra maneira é modificar o operador de seleção, por exemplo, empregando indivíduos elite.

Enquanto, estas garantias são importantes é necessário ressaltar que elas somente são concretizadas para tempos finitos de computação que podem não ser justificáveis ou realizáveis na prática.

Em aplicações onde se deseja obter uma resposta relativamente boa dentro de um intervalo de tempo aceitável, torna-se importante analisar os efeitos das taxas de mutação e

cruzamento e da metodologia de geração da população inicial na qualidade das respostas obtidas.

Torna-se importante avaliar a variância dos resultados de *fitness* obtidos ao final de múltiplas execuções do AG para se ter uma medida da sensibilidade às condições iniciais.

Com isso também se obtém um caso base para se comparar os efeitos da mudança das taxas de cruzamento e/ou mutação. Testes estatísticos de hipóteses podem ser empregados para avaliar se o efeito é significativo.

4.3.4 Considerações sobre Desempenho

AG's constituem-se em problemas computacionais altamente paralelos [6]. Isto os torna aplicáveis a uma extensa faixa de problemas. No entanto, a sua implementação tem que lidar com os problemas inerentes a computações paralelas. Entre eles o mais relevante a dificuldade de implementação de um algoritmo que seja correto, performante e adaptável.

Nos sistemas sem paralelismo explícito, destacam-se dois fatores que afetam a eficiência do uso dos recursos computacionais: geração de números aleatórios e a reiterada avaliação da função adequação com o mesmo cromossomo.

Geradores de números aleatórios

Todos os métodos algorítmicos de geração de números aleatórios necessariamente são exatamente reproduzíveis. Em virtude disto, estes algoritmos são apropriadamente denominados geradores de números pseudoaleatórios.

Basicamente, eles produzem sequências de números que tem um período de repetição extremamente longo. Além disso, uma vez que se conhece a semente (*seed*) do gerador é possível se determinar todos valores da sequencia.

A maior parte dos sistemas operacionais que fornecem geradores aleatórios e a quase

totalidade das bibliotecas padrão de diversas linguagem de programação, quando não explicitamente dirigidas, usam como semente o tempo indicado pelo relógio do sistema.

As qualidades estatísticas dos diversos geradores, quanto à proximidade de uma sequência verdadeiramente aleatória apresenta uma variância significativa. No entanto, a tendência entre os melhores algoritmos é que quanto melhor a aleatoriedade maior o custo computacional.

Via de regra os geradores intermediários são utilizados por padrão. Aplicações que exigem uma maior aleatoriedade utilizam geradores melhores, especialmente criptografia.

Outra metodologia consiste em coletar dados de vários sensores conectados ao computador formando uma reserva de entropia. O sistema operacional (SO) mantém uma estimativa da entropia em bits disponível. Quando um número aleatório de determinada largura de bits é solicitado ao SO se a reserva de entropia for suficiente o SO retorna imediatamente para o processo que fez a requisição. Caso contrário, o SO operacional bloqueia o processo até a reserva acumular a entropia necessária para atender a largura requisitada.

No outro extremo do espectro estão geradores em *hardware*. Normalmente, eles são capazes de fornecer números verdadeiramente aleatórios em grandes larguras de banda. Apresentam custo elevado pois são voltados para aplicações de criptografia que exigem alta segurança.

Cache

O AGC está sujeito a causar a computação da função objetivo com o mesmo argumento repetidas vezes em virtude do seu próprio funcionamento. Entretanto, a complexa interação entre a exploração de vizinhanças através de cruzamento e a aleatoriedade das mutações torna a previsibilidade da frequência de repetição problemática. Consequentemente, torna-se difícil conceber uma otimização que tenha efeitos mensuráveis e consistentes.

Contudo, dada a impossibilidade do AGC atingir o ótimo global [41], torna-se interessante considerar este dispêndio de recursos computacionais na técnica que emprega indivíduos elite.

A otimização mais simples seria além de armazenar o cromossomo dos indivíduos elite armazenar também os resultados da computação da função *fitness*. Contudo, mesmo que o custo de avaliação do grau de adequação seja alto, o ganho de desempenho dependerá de quão cedo o algoritmo convirja.

Além disso, ela possui a característica indesejada de ser mais efetiva quanto maior for o número de gerações que o AG iterar sem melhoria de *fitness*.A filosofia da otimização é válida, apenas a implementação anteriormente citada que deixa muito a desejar.

No âmbito de arquitetura de computadores existe um problema que emprega a mesma filosofia com técnicas de implementação consolidadas. Neste ambiente, caches são empregadas para diminuir o impacto da latência de acessos a memória principal [42].

Os caches neste caso encontram-se em *hardware*. Pelo fato dos padrões de acesso a memória serem dependentes do tipo de código sendo executado o dimensionamento requer extensiva análise, principalmente em virtude de interações com demais parâmetros da arquitetura [43].

No contexto de linguagens de programação, o processo de armazenar resultados anteriores de chamadas a uma função de custo computacional relevante é denominado memoização (*memoization*) [44]. Esta técnica surgiu e é mais amplamente empregada em linguagens com ênfase na disciplina funcional.

Tais linguagens utilizam memoização para reduzir o custo computacional do uso de funções recursivas. Em certos tipo de uso, a técnica viabiliza este tipo de utilização. Não apenas sob o aspecto de tempo computacional, mas torna possível o emprego de funções que não poderiam ser anteriormente implementadas.

Como o objetivo primário é promover o emprego de funções recursivas, não existe um foco no dimensionamento da área de armazenamento dos resultados anteriores. A recursividade gera padrões de acesso bem característicos: quanto maior o grau de recursividade, maior será o benefício da memoização. Diante disso, o suporte para esta técnica é beneficialmente automatizado [44].

Pelo fato de AG's não possuírem a regularidade das chamadas de funções recursivas na avaliação da função *fitness*, não se justifica o emprego de uma área de armazenamento que possa conter todos os valores anteriores da função adequação. Especialmente em uma linguagem como FORTRAN, que não possui suporte nativo para *hash tables*.

Assim sendo, a metodologia para dimensionamento e avaliação de desempenho para caches é mais adequada. A principal figura de mérito de um cache é a taxa de acerto (*hit rate*). Ela é a proporção do número de consultas que encontram o valor desejado já presente no cache para o número total de acessos ao cache: (4.5).

$$h_r = \frac{n_{acertos}}{n_{acessos}} \tag{4.5}$$

Quando uma consulta ao cache falha é necessário chamar a função adequação (incorrendo no seu custo computacional). A taxa de falha (*miss rate*) é complementar à taxa de acerto, (4.6).

$$m_r = 1 - h_r = \frac{n_{falhas}}{n_{acessos}} \tag{4.6}$$

O tempo médio de computação de uma função cacheada $\overline{t_{cf}}$ é dado pela equação (4.7). Na qual $\overline{t_c}$ é o tempo médio necessário para realizar uma busca no cache, $\overline{t_f}$ é o tempo médio que a função adequação gasta da sua chamada a seu retorno e ε é parcela do tempo computacional devido a influência de outros fatores.

$$\overline{t_{cf}} = m_r \overline{t_f} + \overline{t_c} + \varepsilon \tag{4.7}$$

Numa implementação de AG utilizando uma linguagem com gerenciamento manual de memória, compilação para código nativo e onde a execução do AG é o único processo em ambiente de usuário que é limitado pela CPU o fator ε é desprezível.

Uma vez que, observadas as condições anteriores, o processo do AG só será interrompido pelo SO durante a execução do agendador de processos (*scheduler*) ou quando o processo do AG estiver bloqueado esperando a conclusão de E/S (operações de entrada/saída).

Num SO contemporâneo, com único processo em espaço de usuário limitado pela CPU (salvo condições patológicas) não ocorrerá contenção no acesso à memória ou demais recursos de E/S. Enquanto o processo do AG não dependa dos resultados de E/S para continuar a sua computação, o SO agendará as E/S's e voltará a executá-lo imediatamente.

Os ciclos de execução do agendador do SO são projetados para interferirem insignificativamente no desempenho de múltiplos processos concorrendo por recursos de CPU e E/S. As possíveis interferências para um único processo são insignificantes quando comparadas aos efeitos dos padrões de acesso a memória do processo do AG e suas interações com os caches da CPU [45, 42].

O tempo $\overline{t_c}$ é o custo computacional intrínseco do cache. Ele depende tanto da estrutura de dados que se utiliza para implementar o cache, quanto do tamanho dele e da política de relocação de valores.

A estrutura de dados mais simples para implementar o cache para um AG de n_v variáveis representadas por números de ponto flutuante é uma matriz de n_l linhas por n_v colunas. O custo computacional deste cache é dado por (4.8). Onde C_{comp} é o tempo requerido para se fazer a comparação entre duas variáveis de ponto flutuante e $\overline{C_{reloc}}$ é o tempo médio necessário para aplicar a política de relocação escolhida.

$$\overline{t_c} = \overline{t_c^{falha} + t_c^{sucesso}} = (n_l n_v C_{comp} + \overline{C_{reloc}}) m_r + \overline{n_l} n_v C_{comp} h_r \tag{4.8}$$

O valor $\overline{n_l}$ é o número médio de linhas que precisam serem acessadas no caso de uma consulta bem sucedida. Ele depende principalmente da política de relocação e, em menor grau, dos padrões de acesso ao cache. É útil analisar o caso extremo $\overline{n_l} = n_l$, para o qual a equação (4.8) torna-se em (4.9).

$$\overline{t_c} = n_l n_v C_{comp} + \overline{C_{reloc}} m_r \tag{4.9}$$

Dentre as políticas de relocação disponíveis optou-se pela LRU (*Least Recently Used*). Ela caracteriza-se por expulsar do cache a linha que teve o menor número de acessos. De acordo com [46], quando comparado ao algoritmo ótimo de relocação de cache, a maior parte das falhas da LRU são do tipo mais-recentemente-descartado. Nota-se que o algoritmo ótimo de relocação necessita de conhecimento dos acessos futuros [46], não sendo implementável.

O algoritmo ótimo de relocação consiste em descartar a linha de cache que não será utilizada pelo maior período de tempo no futuro. A sua otimalidade foi demonstrada por [47] e, em virtude disso, também é denominado algoritmo de Bélády. Também é conhecido como o algoritmo clarividente.

Apesar de [46] sugerir modificações para a política LRU para aproximar ainda mais o seu desempenho do algoritmo de Bélády, optou-se por implementar a LRU clássica da maneira mais simples possível.

Tal como implementado, o custo de relocação médio da LRU é dado pela equação (4.10), na qual c_1 e c_2 são constantes.

$$\overline{C_{reloc}} = c_1 n_l C_{comp} + c_2 \tag{4.10}$$

O quanto mais rápido o cache LRU é que a alternativa de sempre avaliar a função adequação é dado por (4.11).

$$speedup = \frac{\overline{t_f}}{\overline{t_{cf}}} = \frac{\overline{t_f}}{m_r\overline{t_f} + n_l n_v C_{comp} + c_1 n_l C_{comp} m_r + c_2 m_r} \tag{4.11}$$

Dado que as constantes C_{comp}, c_1 e c_2 podem ser desprezadas, por si só, em computadores contemporâneos e a escolha adequada de um tamanho de cache que propicie uma taxa de falha aceitável, o *speedup* pode ser aproximado para (4.12).

$$speedup \approx \frac{\overline{t_f}}{m_r\overline{t_f}} = \frac{1}{m_r} \tag{4.12}$$

A implementação do AG com indivíduos elite, codificação em ponto flutuante e usando cache LRU encontra-se no Apêndice D.

5 Metodologia

Uma vez definidas as funções objetivos adequadas para cada caso (com ou sem realimentação de fase) procedeu-se uma fase preliminar de execuções dos Algoritmos Genéticos. Nesta fase, foram definidos os parâmetros dos AG's, como tipo de representação genética, taxas de cruzamento e mutação, pesos componentes da função custo. Entre outros, também foram definidos os critérios de parada e tamanho da população.

Quando se verificou respostas satisfatórias, os parâmetros dos AG's foram fixados e se procedeu uma caracterização mais rigorosa das respostas. Para tanto, os AG's foram executados repetidamente e as melhores respostas coletadas. O número de repetições foi selecionado para ser o maior possível, limitado ao máximo de duas horas para serem completadas. Nesta etapa foram feitos alguns ajustes mais finos.

Optou-se por caracterizar o desempenho e a consistência das respostas dos AG's analisando os casos extremos (as respostas com melhor e pior valores da função de adequação) e intermediário. Na definição do caso intermediário, tomou-se a resposta cujo valor de adequação estava mais próximo do valor médio de adequação obtido durante as execuções.

5.1 Funções Objetivo

As respostas do ângulo de potência com e sem realimentação de fase são qualitativamente diferentes. Comparando as Figuras 2.2 (p. 13) e 2.3 (p. 14), observa-se que o ângulo de carga é crescente até a ocorrência do seu valor máximo no caso sem realimentação

($K_d = 0$).

Esta propriedade é válida para todos os casos sem realimentação de fase. Este fato pode ser estabelecido constatando que todas as soluções aceitáveis da equação diferencial (2.1) são nulas para períodos de tempo suficientemente grandes[1]. Consequentemente, todas as características dinâmicas da solução são decorrentes das condições iniciais.

Assim sendo, para determinar o crescimento ou decrescimento da solução é preciso avaliar apenas a condição inicial $\Delta\delta(t)|_{t=0}$. Supondo que o inversor esteja em paralelo com a rede no instante inicial, têm-se que $P(t)|_{t=0^-} = 0$. Logo,

$$\begin{aligned}\Delta\delta(t)|_{t=0} &= -K_d\,\Delta P(t)|_{t=0} + \Delta\delta(t)|_{t=0^-} \\ &= -K_d\,(0 - P_e) + \Delta\delta(t)|_{t=0^-} \\ &= K_d P_e - \delta_e.\end{aligned}$$

Empregando a equação (2.2) obtêm-se a primeira condição inicial: (5.1). Portanto, para todas as soluções aceitáveis e não triviais sem realimentação de fase têm-se, necessariamente, uma resposta crescente entre o instante inicial e o tempo onde o ângulo de potência atinge seu valor máximo.

$$\delta(t)|_{t=0} = K_d P_e \tag{5.1}$$

A equação (5.1) ainda permite concluir que, para o caso com realimentação de fase, o crescimento ou decrescimento da resposta depende de K_d. Considerando que o valor do fluxo de potência ativa em regime permanente seja fixo.

Além disso, os efeitos da oscilação na faixa de passagem dos filtros utilizados na medi-

[1] Equivalentemente, todas as soluções fisicamente relevantes respeitam a condição de contorno estabelecida pela equação (2.11).

ção da potência não podem ser desprezados. Tais efeitos são aparentes na Figura 2.6 (p. 16), estando presentes tanto no modelo de espaço de estados como na simulação do PSIM.

Deste modo, é recomendável empregar funções objetivos bem definidas para as duas classes de resposta ou utilizar funções específicas para cada uma. A primeira alternativa é a mais genérica: tornando o espaço de busca mais complexo em virtude de sua menor capacidade de distinção entre diferentes respostas.

O emprego de funções distintas permite uma maior distinção entre respostas e, consequentemente, uma otimização mais efetiva. Isto é verdadeiro não só para meta-heurísticas, mas para qualquer procedimento de busca.

Adicionalmente, para o sistema de controle sendo otimizado, a separação em duas classes é natural — uma vez que a realimentação de fase foi sugerida como melhoria à técnica básica [15, 48].

No contexto de respostas quase-ótimas, é relevante avaliar os compromissos entre as soluções com e sem realimentação de fase.

5.1.1 Controle sem Realimentação de Fase

Em virtude do caráter crescente da resposta do ângulo de carga, quando K_d é nulo, as métricas clássicas de desempenho de Teoria de Controle podem ser empregadas como funções objetivo. Especificamente, serão consideradas a sobre-resposta (*overshoot*), o tempo de subida (*rise time*) e o tempo de assentamento (*settling time*).

É relevante observar que mesmo utilizando o Modelo de Pequenos Sinais não existem fórmulas analíticas genéricas para expressar as referidas métricas. O caminho mais simples seria aproximar o sistema de terceira ordem para um sistema de segunda ordem.

Esta constitui uma das vantagens do uso de AG's. A função adequação pode ser o resultado de uma simulação numérica, sem necessidade de ser expressável analiticamente ou ser

linear, contínua ou diferenciável.

As métricas que se seguem pressupõem que a função $y(t)$ possa ser calculada para todo t no intervalo $[0,T]$. Além disso, $y(t)$ atinge o seu valor de regime permanente y_e em um tempo menor que T e $y(0) < y_e < \max y(t)$.

Assim sendo, a sobre-resposta de $y(t)$ é dada por (5.2).

$$S_y = \frac{\max y(t)}{y_e} \tag{5.2}$$

O tempo de subida é dado por (5.3), ou seja, ele é o menor tempo no qual $y(t)$ atinge um valor maior ou igual a uma dada fração do valor de regime permanente. No presente trabalho, utilizou-se $f_{ts} = 0,9$.

$$t_s^y = \min\{t' \in [0,T] \ : \ y(t') \geq f_{ts} y_e\} \tag{5.3}$$

O tempo de assentamento é definido por (5.4). Ele é o tempo necessário para que a variação de $y(t)$ fique restrita a uma dada faixa de valores em torno do valor de regime. Particularmente, foi utilizado $f_{ta} = 0,05$.

$$t_a = \min\{t' \in [0,T] \ : \ |y(t') - y(t)| \leq f_{ta} y_e, \ \forall t, t > t'\} \tag{5.4}$$

A função *fitness* para o caso sem realimentação de fase é dada pela equação (5.5). Ela expressa que se deseja minimizar o tempo de subida e a magnitude da sobre-reposta. Em virtude dos efeitos indesejáveis da sobre-resposta (instabilidade, redução de confiabilidade e vida útil do sistema) a sua minimização é priorizada, atribuindo-lhe um peso maior na função objetivo.

$$\mathit{fitness}(K_p, K_v, \omega_f) = 2\,S_\delta[\%] + t_s^\delta[ms] = 200\,S_\delta + t_s^\delta 10^3 \tag{5.5}$$

5.1.2 Controle com Realimentação de Fase

A função objetivo para este caso tem que levar em conta que a resposta do ângulo de potência pode ser tanto crescente como decrescente. Adicionalmente, o valor inicial pode estar arbitrariamente próximo do valor de regime permanente, conforme pode ser inferido da equação (5.1).

Desta forma, seria desejável que a função objetivo favorecesse respostas que não apenas começassem próximas do regime permanente, mas também permanecessem relativamente próximas durante suas oscilações transitórias.

Uma métrica de desempenho que atende estes requisitos é a Integral do Erro Absoluto (IEA), dada por (5.6). Ela também possui origem na Teoria de Controle e pode ser considerada como a base da família de métricas associadas ao erro absoluto. Dentre elas, destaca-se o uso de fatores de ponderação do erro absoluto dependentes do tempo.

$$\mathrm{IEA}(y, T) = \int_0^T |y(t) - y_e|\,\mathrm{d}t \tag{5.6}$$

Continuando o emprego da notação estabelecida anteriormente, em (5.6) y_e representa o valor em regime permanente de $y(t)$ e ele é alcançado em um intervalo de tempo estritamente menor que a largura T da janela de tempo sendo considerada.

A função objetivo utilizada no caso com realimentação de fase é dada por (5.7). T foi fixado em 1 segundo, pois tempos de transferência acima deste valor são indesejáveis.

$$\mathit{fitness}(K_d, K_p, K_v, \omega_f) = \mathrm{IEA}(\delta, 1) \tag{5.7}$$

6 Resultados

O ponto de operação para a qual o AG foi empregado para otimizar a resposta do fluxo de potência ativa encontra-se na Tabela 6.1. A frequência da rede elétrica foi adotada como 60 Hz e a impedância de conexão como $0,5 + j3,44\,\Omega$.

Tabela 6.1: Ponto de operação do sistema.

Variável	Valor	Unidade
δ_e	$0,1558$	rad
P_e	$511,69$	W
Q_e	$80,39$	var
E_e	$107,11$	$V(RMS)$
V_e	$103,40$	$V(RMS)$

Os valores dos parâmetros de controle foram limitados de acordo com a Tabela 6.2. Valores fora destas faixas levam a respostas instáveis, que aumentariam o espaço de busca desnecessariamente. Esta limitação é amparada por resultados práticos e mesmo pelo modelo quando são levados em conta os erros de medição da corrente e da tensão.

Tabela 6.2: Restrições aos valores dos genes.

	Valor	
Variável	Mínimo	Máximo
K_d (rad/W)	$0,0$	$3,0 \times 10^{-4}$
K_p (rad/J)	$0,0$	$1,0 \times 10^{-1}$
K_v (V/var)	$0,0$	$1,7 \times 10^{-2}$
ω_f (rad/s)	$1,2$	$1,0 \times 10$

Foram considerados dois casos, com e sem realimentação de fase. Quando não existe realimentação da fase o ganho K_d é nulo.

6.1 Resultados de Simulação

6.1.1 Sem Realimentação de Fase

A população continha 20 indivíduos dos quais os dois melhores passavam para a geração seguinte sem sofrer alteração. O esquema de codificação das variáveis cromossômicas empregado foi números em ponto flutuante, com taxa de cruzamento de 80% e mutação gaussiana. O critério de parada foi o número máximo de gerações (200). A função adequação, dada por (5.5), foi a soma do dobro da magnitude da sobre-resposta, dada em porcentagem, com o tempo de subida, dado em ms.

O AG foi executado 50 vezes e os resultados encontram-se na Tabela 6.3. Nota-se uma convergência dos parâmetros de controle, com variações que correspondem a diferente composições de sobre-resposta e tempo de subida.

Tabela 6.3: Resultados obtidos em 50 execuções do AG para K_d nulo.

Fitness	Cromossomo			Características da Resposta	
	$K_p\ (rad/Ws)$	$K_v\ (V/var)$	$\omega_f\ (rad/s)$	sobre-resposta $(\%)$	tempo de subida (ms)
Máximo	$0,0012$	$0,0166$	$9,74$	$2,95$	$301,28$
Intermediário	$0,0014$	$0,0140$	$9,99$	$4,78$	$253,91$
Mínimo	$0,0021$	$0,0170$	$10,00$	$15,42$	$148,88$

A Figura 6.1 compara a resposta do ângulo de carga para os parâmetros obtidos pela sintonia por especialista em [15] e os parâmetros obtidos pelo AG. O AG reduziu a magnitude da sobre-resposta de 136,48% para 4,78% e aumentou o tempo de subida de 25,20 ms para 253,91 ms. A Figura 6.2 mostra o fluxo de potência ativa para ambos conjuntos de parâmetros. Observa-se que o incremento no tempo de subida provocado pelo AG é acompanhado pela diminuição do tempo de assentamento.

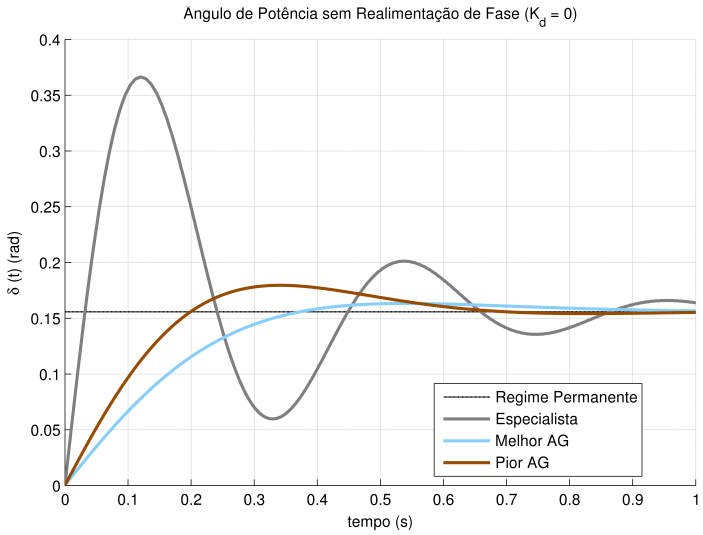

Figura 6.1: Resposta do ângulo de carga usando sintonia por AG e por especialista ($K_d = 0$).

6.1.2 Com Realimentação de Fase

Diferentemente do AG usado para o caso sem realimentação de fase, foi usado apenas um indivíduo elite e a taxa de mutação foi mantida constante em 10%. A função adequação, dada por (5.7), foi a integral da diferença entre o valor do ângulo de carga e o seu valor de regime permanente. Os resultados obtidos encontram-se na Tabela 6.4.

Tabela 6.4: Resultados obtidos para 20 execuções do AG.

Fitness	Cromossomo				Erro Absoluto	
	$K_d\ (rad/W)$	$K_p\ (rad/Ws)$	$K_v\ (V/var)$	$\omega_f\ (rad/s)$	Máximo (rad)	Médio (rad)
Máximo	$0,0003$	$0,0031$	$0,0175$	$9,86$	$4,50 \times 10^{-3}$	$1,54 \times 10^{-3}$
Intermediário	$0,0003$	$0,0024$	$0,0167$	$7,71$	$4,19 \times 10^{-3}$	$1,10 \times 10^{-3}$
Minimo	$0,0003$	$0,0008$	$0,0044$	$2,52$	$2,46 \times 10^{-3}$	$1,50 \times 10^{-3}$

A consequência mais evidente do uso do IEA como função de adequação é a tendência do AG de diminuir a frequência de corte dos filtros passa-baixas usados na medição da potência. Esta redução traduz-se na diminuição da oscilação presente na banda passante, contribuindo significativamente para redução do erro absoluto.

Os resultados obtidos usando AG e pela sintonia por especialista são apresentados nas Figuras 6.3 e 6.4. O AG reduziu o máximo erro absoluto de 0,3585 rad para $2,46 \times 10^{-3}$

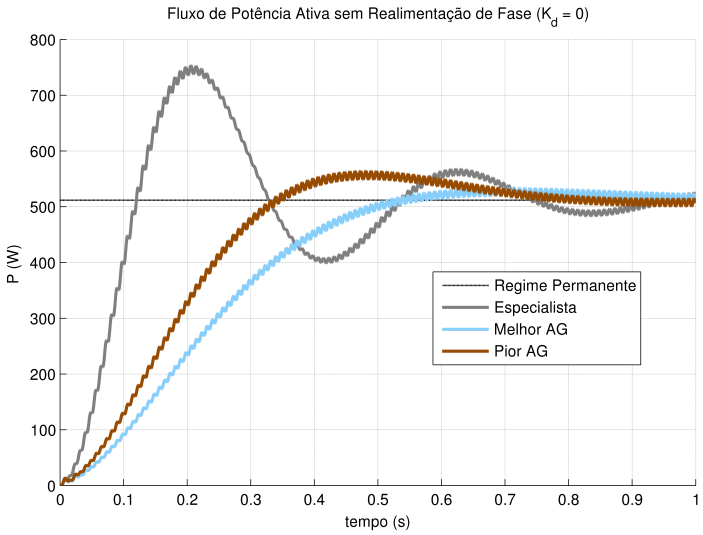

Figura 6.2: Resposta do fluxo de potência ativa usando sintonia por AG e por especialista ($K_d = 0$).

rad quando comparado com a sintonia por especialista. Comparando as Figuras 6.1 e 6.3 é possível perceber a melhoria introduzida pela realimentação do ângulo de carga.

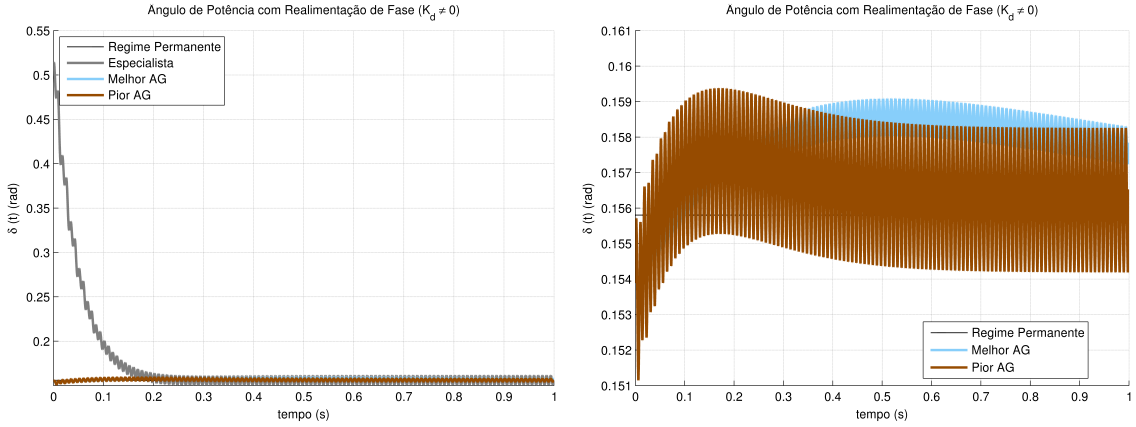

Figura 6.3: Resposta do ângulo de carga usando sintonia por AG e por especialista ($K_d \neq 0$).

6.2 Resultados Experimentais

Os resultados experimentais foram obtidos empregando um inversor monofásico composto por IGBT's em uma configuração ponte-completa [49]. A frequência de chaveamento foi fixada em 18 kHz.

A impedância de conexão consistiu num transformador monofásico com relação 1:2. A

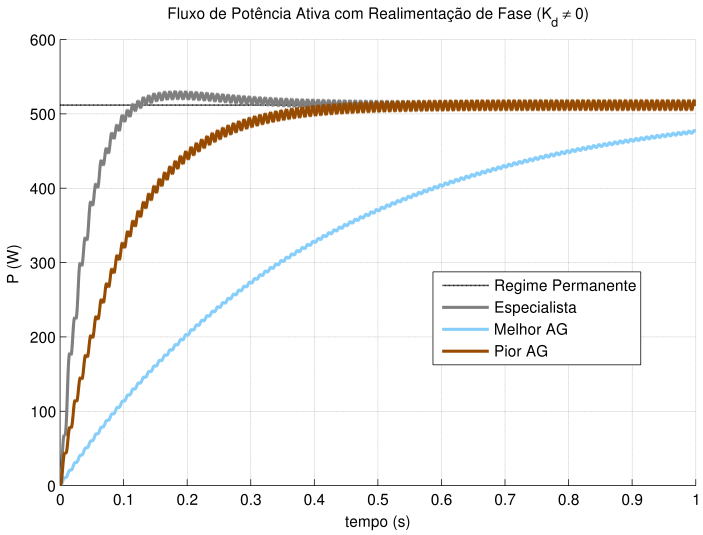

Figura 6.4: Resposta do fluxo de potência ativa usando sintonia por AG e por especialista ($K_d \neq 0$).

sua indutância e resistência, referidas ao lado do inversor, foram, respectivamente, 9,12 mH e 0,5 Ω.

O controle do fluxo de potência foi realizado digitalmente por um computador através de uma placa de aquisição com frequência de amostragem de 5 kHz e resolução de 12 bits na conversão dos sinais analógicos para digitais.

Havia uma chave operada manualmente entre o inversor e a rede. Quando ela era fechada um PLL (*Phase Locked Loop*) digital começava a funcionar no computador, sincronizando a referência interna de controle à tensão da rede. Entretanto, os sinais de chaveamento para o inversor só eram habilitados após 0,1 segundos. No mesmo instante, os referidos sinais passavam a ser controlados conforme a metodologia P–ω e Q–V.

Tanto no caso com realimentação de fase, como no sem, foram levantadas as respostas dos fluxos de potência ativa e reativa utilizando os parâmetros de controle obtidos pelo melhor e o pior resultado do AG. Também foram levantados os fluxos para os parâmetros de controle utilizados em [15], os quais foram designados “Especialista”.

6.2.1 Sem Realimentação de Fase

Nas Figuras 6.5 e 6.6 observam-se, respectivamente, os fluxos ativo e reativo obtidos pelo AG e sintonia por especialista. Nota-se que a melhor resposta (segundo a simulação) para o AG não atinge o valor de regime permanente para a potência ativa dentro da janela de tempo do experimento.

O primeiro motivo para isto é o que AG não otimiza diretamente o fluxo de potência ativa, mas o ângulo de potência, cuja resposta dinâmica é mais rápida. Contudo, o maior atenuador do ângulo de potência é a impedância de conexão e sua influência não é o suficiente para explicar a discordância com a simulação.

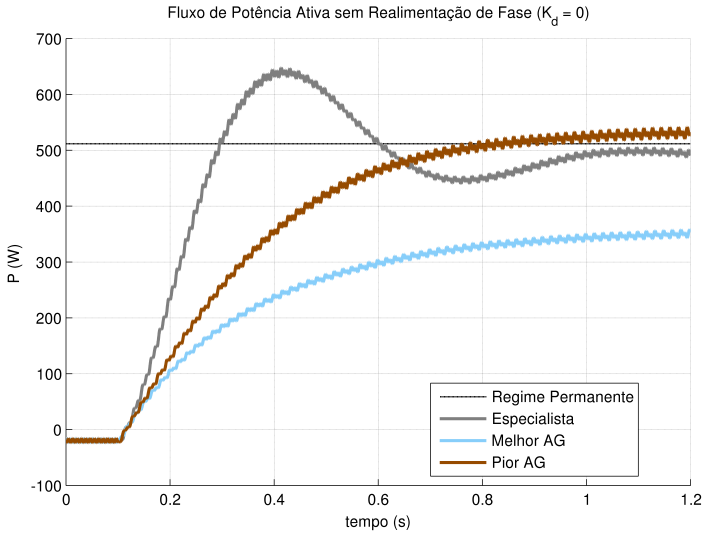

Figura 6.5: Fluxo de potência ativa experimental usando sintonia por AG e por especialista ($K_d = 0$).

Na Figura 6.6 constata-se que nenhuma das sintonias dos parâmetros de controle atingiu o valor previsto de regime permanente do reativo. Além disso, o valor inicial da potência reativa (aproximadamente 300 var) é consideravelmente diferente do valor considerado nas simulações com o AG (0 var). Estes fatos são suficientes para explicar a divergência entre o ativo experimental e simulado para melhor sintonia do AG.

Comparando as Figuras 6.2 e 6.5 observa-se que as respostas experimentais são mais atenuadas que as simuladas. Enquanto não é possível eliminar o efeito das diferentes con-

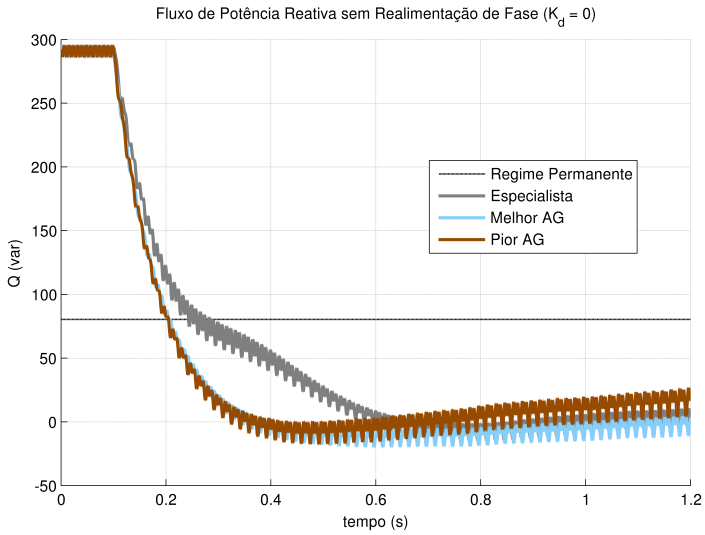

Figura 6.6: Fluxo de potência reativa experimental usando sintonia por AG e por especialista ($K_d = 0$).

dições iniciais do reativo, é razoável considerar que uma parte da atenuação é devida ao efeito de resistências no aparato experimental que não foram completamente modeladas nas simulações.

Ainda com relação à Figura 6.6, é preciso considerar o efeito dos erros de medição da tensão do inversor na regulação do reativo. O reativo na metodologia de controle por curva Q–V é diretamente proporcional à variação da tensão de saída do inversor em relação ao seu valor de regime permanente ($\Delta E = K_v \Delta Q$). Assim sendo, para $K_v = 0,01$, um erro de medição de 1 V resulta num erro de 100 var.

6.2.2 Com Realimentação de Fase

Nas Figuras 6.7 e 6.8 encontram-se, respectivamente, as respostas dos fluxos ativo e reativo coletadas para as sintonias por AG e especialista. Como no caso sem realimentação de fase, o valor previsto de regime para a potência reativa não foi alcançado para nenhuma das sintonias e o valor inicial também apresentou uma diferença considerável com relação ao empregado nas simulações.

Em contraste com o caso anterior, nenhuma das respostas do AG na Figura 6.7 atingiu o

valor previsto de regime. A sintonia por AG com pior resultado de simulação atingiu valor de regime permanente 20% menor que o esperado, enquanto o AG com melhor função custo não alcançou regime permanente dentro da janela temporal do experimento.

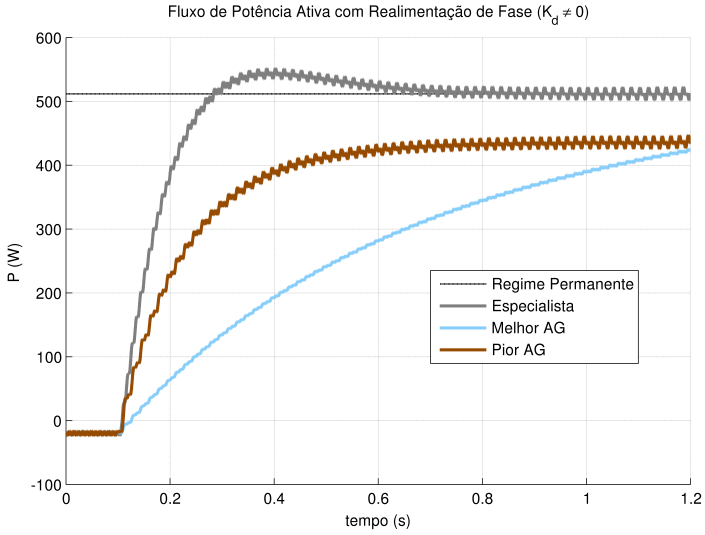

Figura 6.7: Fluxo de potência ativa experimental usando sintonia por AG e por especialista ($K_d \neq 0$).

Na Figura 6.8 nota-se, diferentemente do constatado para o caso sem realimentação, que nenhuma das respostas superou a fase transitória dentro do tempo de ensaio.

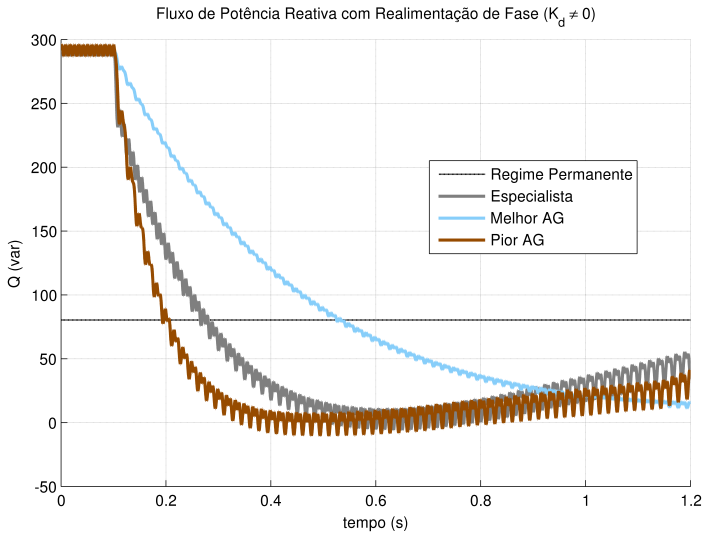

Figura 6.8: Fluxo de potência reativa experimental usando sintonia por AG e por especialista ($K_d \neq 0$).

7 Conclusões

A otimização da resposta do ângulo de potência utilizando Algoritmos Genéticos permitiu a obtenção de melhores respostas simuladas que as sintonias por especialista, tanto no caso sem realimentação como com realimentação de fase. No primeiro caso, a sobre resposta foi diminuída de 136,2% para 4,78% e o tempo de subida passou de 25,2 ms para 253,91 ms. No último, o máximo erro absoluto foi reduzido de 0,3585 rad para $2,46 \times 10^{-3}$ rad e o erro absoluto médio de 0,0195 rad para $1,50 \times 10^{-3}$ rad.

No caso com K_d nulo, apesar do aumento do tempo de subida necessário para redução da sobre resposta, houve uma melhoria significativa do tempo de assentamento: de 799,68 ms para 329,23 ms.

Os resultados experimentais confirmaram, para o caso sem realimentação, a obtenção de uma melhor resposta pelo AG que pela sintonia por especialista. Contudo, destacaram-se a diferença das respostas da potência reativa com relação aos resultados simulados. Em primeiro lugar, quanto ao valor inicial; em segundo, quanto ao valor de regime permanente.

O primeiro problema é facilmente tratável: basta ajustar as condições iniciais dos modelos de simulação para àquelas observadas nos experimentos. Já o segundo, precisa de uma abordagem tanto experimental como uma revisão de modelo.

No aspecto experimental, é importante reduzir os erros de medição da tensão do inversor o máximo possível. Adicionalmente, como a corrente do inversor é uma das entradas para o controle, a possibilidade de diminuição dos erros da sua medição também merece ser

estudada. O principal fator limitante no experimento foi a baixa taxa de amostragem (5 kHz) da placa de aquisição disponível. Um sistema de aquisição mais capaz possibilitaria utilizar toda a resolução do conversor analógico-digital e, empregando super amostragem, tornar o controle mais robusto a presença de ruídos através da aplicação de processamento digital de sinais.

Quanto ao modelo, seria necessário levar em conta pelo menos o erro de medição da tensão. Uma vez que o modelo utilizado já é discreto não seria difícil adaptá-lo. Entretanto, dada a flexibilidade do AG, talvez a solução mais natural seria modificar as funções objetivos. Qualquer que seja o caminho adotado, o desafio é obter um modelo de erros que seja adequado e tenha um custo computacional razoável.

7.1 Modelo em Espaço de Estados

O modelo de espaço de estados utilizado pelo AG na determinação do valor da função objetivo mostrou um bom equilíbrio entre fidelidade, simplicidade e custo computacional. No primeiro aspecto, ele mostrou-se mais realista que o modelo obtido por Análise de Pequenos Sinais por dois motivos: não apresenta o erro de linearização das funções trigonométricas; modela as oscilações presentes nas bandas de passagem dos filtros passa-baixas.

Quanto à simplicidade, ele é o modelo não-linear mais simples possível para o sistema sob estudo: todos os seus subcomponentes são modelos lineares. Uma vantagem adicional da simplicidade (além de facilidade de implementação) é o baixo custo computacional. Isto é especialmente importante pelo fato do AG necessitar fazer um grande número de simulações do sistema.

7.2 Algoritmos Genéticos

Os AG's mostraram-se tanto uma metologia consolidada como versátil. O que é evidenciado pela extensiva literatura, abrangendo diversas áreas de aplicação e pelos sucessivos refinamentos que foram sugeridos. Contudo, apenas a introdução de indivíduos elite ao AGC foi suficiente para se obter os resultados apresentados no presente trabalho.

Enquanto, especialmente o caso sem realimentação de fase, poderia se beneficiar de adaptações voltadas para solução de funções multi-objetivos, os resultados obtidos através de múltiplas execuções mostrou um bom compromisso entre consistência e variabilidade das respostas.

Quanto a metodologia de codificação dos cromossomos, ainda não existe consenso quanto qual é a melhor forma de representação, binária de comprimento fixo ou ponto flutuante. Apenas conhecem-se aplicações particulares nas quais uma técnica apresenta melhor desempenho. Assim sendo, a escolha entre elas deve se basear nos diferentes compromissos que elas implicam. Entretanto, a única forma assegurada de obter o melhor desempenho é fazer testes comparativos. Enquanto existem formas mais elaboradas de representação genômica e/ou fenotípica, as suas características são normalmente conhecidas somente para casos particulares. Soma-se a isso a tendência de serem computacionalmente mais complexas e mais recentes, tendo menos resultados publicados.

O emprego de caches para melhoria do desempenho computacional mostrou-se uma otimização significante. Entretanto, em virtude do seu baixo custo em comparação com a avaliação da função adequação, a abordagem mais eficiente globalmente foi implementar um cache grande o suficiente para conter todos os cromossomos passíveis de serem produzidos durante a execução do AG. Porém, observa-se que isto foi possibilitado porque não havia contenção significante para uso de memória. Em situações com disponibilidade limitada de memória, o uso de cache como uma política de relocação é imprescindível. Mesmo quando

não ocorrem limitações de espaço, o uso marginalmente mais eficiente da memória só é alcançado através de relocação.

7.3 Trabalhos Futuros

Nos casos estudados, sempre utilizou-se AG's para otimizar a resposta do ângulo de carga e, consequentemente, o fluxo de potência ativa. Seria relevante otimizar diretamente o fluxo. Em virtude de diferentes características das respostas destas variáveis, as métricas de desempenho utilizadas na elaboração da função objetivo teriam de ser reconsideradas. Intuitivamente, esperaria-se que os AG's obtivessem melhores resultados nesta abordagem direta. Entretanto, os resultados obtidos num estudo preliminar ainda não publicado apontam na direção contrária.

No caso sem realimentação de fase, poderia se avaliar o desempenho do AG utilizando somente o tempo de assentamento como função de adequação. Adicionalmente, seria importante aplicar um AG adaptado para funções multi-objetivos, especialmente o NSGA-II, na otimização simultânea da sobre-resposta e do tempo de subida. Desta forma, poderia se caracterizar quais as vantagens de funções multi-objetivos, tanto na sua implementação mais simples quanto na de melhor desempenho, frente a uma função com único objetivo.

Os resultados experimentais mostraram a importância dos erros de medição, particularmente da tensão de saída do inversor, no desempenho e viabilidade do controle por curvas P–ω e Q–V. Assim sendo, é necessário incorporar ao modelo de simulação de espaço de estados estes efeitos. Também é preciso investigar a necessidade de mudanças nas funções *fitness* e adaptações ao AG.

Referências Bibliográficas

[1] CHANDORKAR, M. C. *et al.* "*Novel architectures and control for distributed UPS systems*". In: *APEC'94*. [S.l.: s.n.], 1994. p. 683–689.

[2] TULADHAR, A. *et al.* "*Parrallel operation of single phase inverter modules with no control interconnections*". In: *APEC'97*. [S.l.: s.n.], 1997. v. 1, p. 94–100.

[3] COELHO, Ernane Antônio Alves. "Técnicas de Controle Aplicadas ao Paralelismo de Inversores". Tese (Tese de doutorado) — Universidade Federal de Minas Gerais, 2000.

[4] ZHOU, K.; DOYLE, J.C.; GLOVER, K. "Robust and optimal control". [S.l.]: Prentice Hall, 1996. ISBN 9780134565675.

[5] CHIANG, S.J.; CHANG, J.M. "*Parallel control of the ups inverters with frequency-dependent droop scheme*". In: *Power Electronics Specialists Conference, 2001. PESC. 2001 IEEE 32nd Annual*. [S.l.: s.n.], 2001. v. 2, p. 957–961. ISSN 0275-9306.

[6] DARRELL; WHITLEY. "*An overview of evolutionary algorithms: practical issues and common pitfalls*". *Information and Software Technology*, v. 43, n. 14, p. 817–831, 2001. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584901001884>.

[7] ZITZLER, E.; THIELE, L. "*Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach*". *Evolutionary Computation, IEEE Transactions on*, v. 3, n. 4, p. 257–271, nov 1999. ISSN 1089-778X.

[8] CHEN, Gaobo; CHEN, Xiufang. "*A hybrid of adaptive genetic algorithm and pattern search for stock index optimized replicate*". In: *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on*. [S.l.: s.n.], 2011. p. 4912 –4915.

[9] WEILE, D.S.; MICHIELSSEN, E. "*Genetic algorithm optimization applied to electromagnetics: a review*". In: *IEEE Transactions on Antennas and Propagation*. [S.l.: s.n.], 1997. v. 45, p. 343–353.

[10] KOUMBOULIS, F.N.; TZAMTZI, M.P. "*A metaheuristic approach for controller design of multivariable processes*". In: *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*. [S.l.: s.n.], 2007. p. 1429–1432.

[11] FLEMING, P. J. *et al.* "*Genetic algorithms in control systems engineering*". In: *In Proceedings of the 12th IFAC World Congress*. [S.l.]: Preprints, 2001. p. 383–390.

[12] JONES, D.F; MIRRAZAVI, S.K; TAMIZ, M. "*Multi-objective meta-heuristics: An overview of the current state-of-the-art*". *European Journal of Operational Research*, v. 137, n. 1, p. 1–9, 2002. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com-/science/article/pii/S0377221701001230>.

[13] BANDYOPADHYAY, S. *et al.* "*A simulated annealing-based multiobjective optimization algorithm: Amosa*". *Evolutionary Computation, IEEE Transactions on*, v. 12, n. 3, p. 269–283, june 2008. ISSN 1089-778X.

[14] JAEGGI, D.M. *et al.* "*The development of a multi-objective tabu search algorithm for continuous optimisation problems*". *European Journal of Operational Research*, v. 185, n. 3, p. 1192–1212, 2008. ISSN 0377-2217. Disponível em: <http://www.sciencedirect-.com/science/article/pii/S0377221706006357>.

[15] PAIVA, Élcio Precioso de. "Uma Proposta de Controle de Paralelismo de Inversores com a Rede Elétrica Utilizando-se a Técnica de Realimentação de Fase". Tese (Tese de doutorado) — Universidade Federal de Uberlândia, 2006.

[16] CHEN, Chi-Tsong. "Linear System Theory and Design". New York, NY, USA: Oxford University Press, Inc., 1998.

[17] MAIA, Helder Z.; PINTO, João O. P.; COELHO, Ernane A. A. "*Power response optimization of inverter grid parallel operation using p-ω and q-v curves, and phase feedback based on genetic algorithm*". *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, p. 1679–1684, 2008.

[18] WEISSTEIN, Eric W. "Optimization Theory". Disponível em: <http://mathworld-.wolfram.com/OptimizationTheory.html>.

[19] BOYD, Stephen; VANDENBERGHE, Lieven. "Convex Optimization". New York, NY, USA: Cambridge University Press, 2004. ISBN 0521833787. Disponível em: <http://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf>.

[20] TADDY, Matthew A. *et al.* "*Bayesian guided pattern search for robust local optimization*". *Technometrics*, v. 51, n. 4, p. 389–401, 2009. Disponível em: <http://pubs-.amstat.org/doi/abs/10.1198/TECH.2009.08007>.

[21] BOER, Pieter-Tjerk de *et al.* "*A tutorial on the cross-entropy method*". *Annals of Operations Research*, Springer Netherlands, v. 134, p. 19–67, 2005. ISSN 0254-5330. 10.1007/s10479-005-5724-z. Disponível em: <http://dx.doi.org/10.1007/s10479-005-5724-z>.

[22] BANKS, Alec; VINCENT, Jonathan; ANYAKOHA, Chukwudi. "*A review of particle swarm optimization. part i: background and development*". *Natural Computing*, Springer Netherlands, v. 6, p. 467–484, 2007. ISSN 1567-7818. 10.1007/s11047-007-9049-5. Disponível em: <http://dx.doi.org/10.1007/s11047-007-9049-5>.

[23] ____. "*A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications*". *Natural Computing*,

Springer Netherlands, v. 7, p. 109–124, 2008. ISSN 1567-7818. 10.1007/s11047-007-9050-z. Disponível em: <http://dx.doi.org/10.1007/s11047-007-9050-z>.

[24] MEZURA-MONTES, Efrñn; VELÁZQUEZ-REYES, Jesús; COELLO, Carlos A. Coello. "*A comparative study of differential evolution variants for global optimization*". In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2006. (GECCO '06), p. 485–492. ISBN 1-59593-186-4. Disponível em: <http://doi.acm.org/10.1145/1143997.1144086>.

[25] JAEGGI, Daniel *et al.* "*Multi-objective parallel tabu search*". In: YAO, Xin *et al.* (Ed.). *Parallel Problem Solving from Nature - PPSN VIII*. Springer Berlin / Heidelberg, 2004, (Lecture Notes in Computer Science, v. 3242). p. 732–741. ISBN 978-3-540-23092-2. Disponível em: <http://dx.doi.org/10.1007/978-3-540-30217-9_74>.

[26] MITCHELL, Melanie. "An Introduction to Genetic Algorithms". [S.l.]: MIT Press, 1998.

[27] GOLDBERG, David Edward. "Genetic algorithms in search, optimization, and machine learning". [S.l.]: Addison-Wesley, 1989.

[28] THORNTON, Stephen. "*Karl popper*". In: ZALTA, Edward N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Winter 2011. [S.l.: s.n.], 2011.

[29] HANSSON, Sven Ove. "*Science and pseudo-science*". In: ZALTA, Edward N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Fall 2008. [S.l.: s.n.], 2008.

[30] KUTSCHERA, Ulrich; NIKLAS, Karl J. "*The modern theory of biological evolution: an expanded synthesis*". *Naturwissenschaften*, Springer Berlin / Heidelberg, v. 91, p. 255–276, 2004. ISSN 0028-1042. 10.1007/s00114-004-0515-y. Disponível em: <http://dx.doi.org/10.1007/s00114-004-0515-y>.

[31] SALATHÉ, Marcel *et al.* "*Rapid parasite adaptation drives selection for high recombination rates*". *Evolution*, v. 62, p. 295–300, February 2008.

[32] KARLIN, Samuel. "*Mathematical models, problems, and controversies of evolutionary theory*". In: *Bulletin (New Series) of the American Mathematical Society*. [S.l.: s.n.], 1984. v. 10, n. 2, p. 221–273.

[33] ĆIRKOVIĆ, Milan M. "*On the first anthropic argument in astrobiology*". *Earth, Moon, and Planets*, Springer Netherlands, v. 91, p. 243–254, 2002. ISSN 0167-9295. 10.1023/A:1026266630823. Disponível em: <http://dx.doi.org/10.1023-/A:1026266630823>.

[34] TRUT, Lyudmila; OSKINA, Irina; KHARLAMOVA, Anastasiya. "*Animal evolution during domestication: the domesticated fox as a model*". *BioEssays*, v. 31, p. 349–360, March 2009.

[35] THEOBALD, Douglas L. "*A formal test of the theory of universal common ancestry*". *Nature*, Macmillan Publishers Limited. All rights reserved, v. 465, n. 7295, p. 219–222, May 2010. ISSN 0028-0836. Disponível em: <http://dx.doi.org/10.1038/nature09014>.

[36] STEEL, Mike; PENNY, David. "*Origins of life: Common ancestry put to the test*". *Nature*, Nature Publishing Group, v. 465, n. 7295, p. 168–169, May 2010. ISSN 0028-0836. Disponível em: <http://dx.doi.org/10.1038/465168a>.

[37] GOLDBERG, David. "*What every computer scientist should know about floating-point arithmetic*". *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 23, p. 5–48, March 1991. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/103162.103163>.

[38] WRIGHT, Alden H. "*Genetic algorithms for real parameter optimization*". In: *Foundations of Genetic Algorithms*. [S.l.]: Morgan Kaufmann, 1991. p. 205–218.

[39] PALESI, M.; GIVARGIS, T. "*Multi-objective design space exploration using genetic algorithms*". In: *Hardware/Software Codesign, 2002. CODES 2002. Proceedings of the Tenth International Symposium on*. [S.l.: s.n.], 2002. p. 67 –72.

[40] CANFORA, Gerardo *et al.* "*An approach for qos-aware service composition based on genetic algorithms*". In: *Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2005. (GECCO '05), p. 1069–1075. ISBN 1-59593-010-8. Disponível em: <http://doi.acm.org/10.1145/1068009.1068189>.

[41] RUDOLPH, G. "*Convergence analysis of canonical genetic algorithms*". *Neural Networks, IEEE Transactions on*, v. 5, n. 1, p. 96–101, jan 1994. ISSN 1045-9227.

[42] HENNESSY, John L.; PATTERSON, David A. "Computer Architecture: A Quantitative Approach, 4th Edition". 4. ed. [S.l.]: Morgan Kaufmann, 2006. Paperback. ISBN 0123704901.

[43] SKADRON, K. *et al.* "*Branch prediction, instruction-window size, and cache size: performance trade-offs and simulation techniques*". *Computers, IEEE Transactions on*, v. 48, n. 11, p. 1260 –1281, nov 1999. ISSN 0018-9340.

[44] NORVIG, Peter. "*Techniques for automatic memoization with applications to context-free parsing*". *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 17, p. 91–98, mar. 1991. ISSN 0891-2017. Disponível em: <http://dl.acm.org/citation.cfm?id=971738-.971743>.

[45] DREPPER, Ulrich. "What Every Programmer Should Know About Memory". 2007. Disponível em: <http://people.redhat.com/drepper/cpumemory.pdf>.

[46] PUZAK, Thomas Roberts. "Analysis of Cache Replacement-Algorithms". Tese (Tese de doutorado) — UMass Amherst, 1985. Disponível em: <http://scholarworks.umass.edu-/dissertations/AAI8509594>.

[47] BELADY, L. A. "*A study of replacement algorithms for a virtual-storage computer*". *IBM Systems Journal*, v. 5, n. 2, p. 78–101, 1966. ISSN 0018-8670.

[48] COELHO, E. A. A. *et al.* "*Small signal analysis applied to a single-phase inverter prototype connected to stiff ac system using an improved power controller*". In: *XV Congresso Brasileiro de Automática*. [S.l.: s.n.], 2004.

[49] MAIA, Helder Z.; PINTO, João O. P.; COELHO, Ernane A. A. "*Power response optimization of inverter grid parallel operation using p-ω and q-v curves, and phase feedback based on genetic algorithm*". *Congresso Brasileiro de Eletrônica de Potência, 2007. COBEP'07 - 9th Brazilian Power Electronics Conference*, 2007.

APÊNDICE A – Implementação da dinâmica da impedância de conexão em Espaço de Estados

```
module lt_module

    use config

contains

    subroutine lt(i, u)
    !               R               L
    !   +-------/\/\/\/\--------))))))))----------+
    !   |              ----->              |                                  10
    !   |                i                 |
    !   |                                  |
    !  (Vs)                               (Vr)
    !   |                                  |
    !   |                                  |
    !   +-----------------------------------------+
    !
    ! Realização Espaço de Estados
    !
    ! dx    - R     1                                                        20
    ! -- = - x + - (Vs - Vr)
    ! dt      L     L
    !
    ! y = x
    !
    ! Discretizando ( u[k] = Vs[k] - Vr[k] = Vinv[k] - Vgrid[k] )
    !
    ! x[k + 1] = exp(- R*Ts/L)*x[k] + (1 - exp(- R*Ts/L))/R * u[k]
    ! y[k] = x[k]
    !                                                                        30
    ! Como y[k] = x[k] = i[k]
    !
    ! i[k + 1] = exp(- R*Ts/L)*i[k] + (1 - exp(- R*Ts/L))/R * u[k]
    !
        implicit none

        real, intent (inout) :: i
```

```
        real, intent (in) :: u

        real :: Ad, Bd                                                    40

        Ad = exp(- R*DT/L)
        Bd = (1.0 - Ad)/R

        i = Ad*i + Bd*u
    end subroutine lt

end module lt_module
```

APÊNDICE B – Implementação de filtro passa-baixas em Espaço de Estados

```
module low_pass_filter

    use config

contains

    subroutine low_pass(y, u, wf)
        ! Implementação de filtro Passa-Baixas de Primeira-Ordem
        !
        !            wf                                                  10
        ! tf =   ---------
        !          s + wf
        !
        !  por Espaço     de Estados
        !
        !  dx
        !  -- = - wf * x + u
        !  dt
        !  y  = wf * x
        !                                                                 20
        ! discretizando
        !
        ! x[k + 1] = exp(- wf*Ts )*x[k] + (1 - exp( - wf*Ts ))/wf * u[k]
        ! y[k] = wf*x[k]
        !
        ! observando que x[k] = y[k]/wf e x[k + 1] = y[k + 1]/wf
        !
        ! y[k + 1] = exp(- wf*Ts)*y[k] + (1 - exp(- wf*Ts))*u[k]
        !
        implicit none                                                    30

        real, intent (inout) :: y
        real, intent (in) :: u, wf

        y = exp(- wf*DT)*y + (1 - exp(- wf*DT))*u

    end subroutine low_pass

end module low_pass_filter
```

APÊNDICE C – Código do simulador em Espaço de Estados

```
module simulate

    use config
    use lt_module
    use power_calc_module
    use low_pass_filter

    real, parameter :: MAX_TIME = 0.2

    type metrics                                                    10
        real :: risetime = -1.0
        real :: overshoot = -1.0
        real :: settlingtime = -1.0
        real :: total_error = 0.0
    end type

contains

    function simulate_InvCon(vals) result (metricas)
        implicit none                                               20

        real, intent (in), dimension(4) :: vals

        ! metricas(1) = angle, metricas(2) = power
        type(metrics), dimension(2) :: metricas

        real :: kd, kp, kv, wf

        real :: soma = 0.0
        real :: t = 0.0                                             30
        real :: v_inv, vq_inv, v_grid, i_inv
        real :: delta = 0.0
        type(power) :: potencia
        real :: P, Q

        integer :: n = 0
        integer :: NMAX = MAX_TIME/DT
        real :: e_max_t = E_MAX
        real :: w_t = W
```

```
real :: deltaP, deltaQ
real :: deltaAngle = 0.0

kd = vals(1)
kp = vals(2)
kv = vals(3)
wf = vals(4)

! inicializaçãoes (absolutamente necessárias)
! Parece que o valor utilizado na declaração (p.e. real :: total_error =
! 0.0 ) só é aplicado a primeira vez que a função é chamada.
soma = 0.0
e_max_t = E_MAX
w_t = W
delta = 0.0
deltaAngle = 0.0
P = 0.0
Q = 0.0
i_inv = 0.0

metricas%risetime = (/ -1.0, -1.0 /)
metricas%overshoot = (/ -1.0, -1.0 /)
metricas%settlingtime = (/ -1.0, -1.0 /)
metricas%total_error = (/ 0.0, 0.0 /)

do n = 0, NMAX
    t = n*DT
    v_inv = e_max_t*sin(delta)
    vq_inv = e_max_t*cos(delta)
    v_grid = V_MAX*sin(W*t)
    call lt(i_inv, v_inv - v_grid)
    potencia = power_calc(v_inv, i_inv, vq_inv)

    call low_pass(P, potencia%active, wf)
    deltaP = P - P0
    w_t = - kp*deltaP + W
    soma = soma + w_t*DT ! integrador
    delta = soma - kd*deltaP
    deltaAngle = delta - W*t

    call low_pass(Q, potencia%reactive, wf)
    deltaQ = Q - Q0
    e_max_t = - kv*deltaQ + E_MAX

    call updateMetrics(P, P0, t, 1, metricas)
    call updateMetrics(deltaAngle, DELTA0, t, 2, metricas)
enddo

if (metricas(1)%overshoot > 0.0) then
    metricas(1)%overshoot = 100.0*metricas(1)%overshoot/DELTA0
endif
if (metricas(2)%overshoot > 0.0) then
    metricas(2)%overshoot = 100.0*metricas(2)%overshoot/P0
endif
```

```
end function

subroutine updateMetrics(var, SSval, t, idx, m)
    implicit none

    real, intent(in) :: var, SSval, t                                    100
    integer, intent(in) :: idx
    type(metrics), intent(inout), dimension(2) :: m

    real :: absError

    absError = abs(var - SSval)

    if (m(idx)%overshoot < var) then
        m(idx)%overshoot = var
    endif                                                                110

    if ( (var >= 0.9*SSval) .and. (m(idx)%risetime < 0.0) ) then
        m(idx)%risetime = t
    endif

    if ( absError <= settlingTimePercentage*SSval ) then
        if ( m(idx)%settlingtime < 0.0 ) then
            m(idx)%settlingtime = t
        endif
    else                                                                 120
        m(idx)%settlingtime = -1.0
    endif

    m(idx)%total_error = m(idx)%total_error + absError
end subroutine

subroutine exaustiveSearch(nbits)
    implicit none

    integer, intent(in) :: nbits                                         130

    type(metrics), dimension(2) :: metricas

    real, parameter, dimension(4) :: minn = (/ 0.0, 0.0, 0.0, 1.0 /)
    real, parameter, dimension(4) :: maxx = (/ 5e-4, 0.1, 5e-2, 10.0 /)

    integer :: n
    integer :: i, j, k, l

    real, dimension(4) :: steps                                          140
    real, dimension(4) :: vals

    n = 2**nbits - 1

    steps = (1.0/n)*(maxx - minn)

    vals = minn

    !open(unit = 200, file = 'exaustiveSearch.dat', action = 'WRITE', form = 'FORMATTED', buffered =
```

```
        open(unit = 200, file = 'exaustiveSearch.dat', action = 'WRITE', form = 'FORMATTED', buffered =

        do i = 0, n
            do j = 0, n
                do k = 0, n
                    do l = 0, n
                        metricas = simulate_InvCon(vals)
                        write (200, '(12ES25.16E3)') vals, metricas(1)%risetime, metricas(1)%overshoot, metri
                        vals(4) = vals(4) + steps(4)
                    enddo
                    vals(4) = minn(4)                                                          160
                    vals(3) = vals(3) + steps(3)
                enddo
                vals(3) = minn(3)
                vals(2) = vals(2) + steps(2)
            enddo
            vals(2) = minn(2)
            vals(1) = vals(1) + steps(1)
        enddo

        close(unit = 200)                                                                      170
    endsubroutine

end module

program runSearch
    use simulate
    implicit none

    ! Caso base para DT = 1e-6 => tempo total < 5 s
    !              para DT = 100e-6 => tempo total < 0.07 s                                    180
    !call exaustiveSearch(2)

!real     3m14.257s
!user     3m12.744s
!sys      0m1.320s
    call exaustiveSearch(5)
end program
```

APÊNDICE D – Código do AG utilizando cache

```
module ga_mod

    use ga_options_mod

    type performance_reg
        real :: minn, meann, maxx
    end type performance_reg

    type cromosome
        real :: fitness
        real, dimension(:), pointer :: cromosome => null()
    end type cromosome

    type cached
        integer :: filled_position = 0
        integer :: accesses = 0
        integer :: hits = 0
        real :: time_cache = 0.0
        real :: time_func = 0.0
        real, dimension(:, :), pointer :: cromosomes => null()
        real, dimension(:), pointer :: fitness => null()
        integer, dimension(:), pointer :: crom_hits => null()
    end type cached

    interface
        function fitnessFunc(individual)
            implicit none

            real, dimension(:) :: individual

            real :: fitnessFunc
        end function fitnessFunc
    end interface

    ! Subroutina que recebe como argumento o melhor indivíduo obtido pelo GA
    interface
        subroutine bestIndividual_GA(individual)
            implicit none

            real, dimension(:) :: individual
        end subroutine
    end interface
```

```
    ! Número de cruzamentos e mutações a cada geração
    integer :: ncrosses, nmutations

    type(cached) :: cache

contains

    function run_ga() result (performance)
        implicit none

        type(performance_reg), dimension(ga_opts%ngenerations + 1) :: performance
        type(cromosome) :: best_individual

        real, dimension(ga_opts%population_size, ga_opts%num_vars) :: pop
        real, dimension(ga_opts%population_size) :: fit
        real :: max_fitness

        integer :: gen
        integer, dimension(1) :: max_idx, min_idx
        integer :: nstall = 0

        integer :: i
        integer :: debug_file = 20

        ! aloca espaço para o cache
        allocate(cache%cromosomes(ga_opts%cache_size, ga_opts%num_vars))
        allocate(cache%fitness(ga_opts%cache_size))
        allocate(cache%crom_hits(ga_opts%cache_size))
        cache%cromosomes = 0.0
        cache%fitness = 0.0
        cache%crom_hits = 0
        cache%filled_position = 0
        cache%accesses = 0
        cache%hits = 0
        cache%time_cache = 0.0
        cache%time_func = 0.0

        ! alocação de espaço para o cromossomo no melhor indivíduo
        allocate(best_individual%cromosome(ga_opts%num_vars))

        ! População inicial
        pop = initialize_population()

        ! Determinação do fitness e das estatísticas da população inicial
        fit = fitness(pop)
        performance(1)%minn = minval(fit)
        performance(1)%meann = sum(fit)/ga_opts%population_size
        performance(1)%maxx = maxval(fit)

        ! Determinação do melhor indivíduo da população inicial
        best_individual%fitness = performance(1)%minn
        min_idx = minloc(fit)
        best_individual%cromosome = pop(min_idx(1), :)
```

```
! Cabeçalho do relatório padrão
if (ga_opts%print_progress) then
    write (*, '(a5, 2a11, 3a16, 2a15, 2a11, 4a16)') &                    100
        '%gen', 'ncrosses', 'nmutations', 'fit_min', 'fit_mean', 'fit_max', &
        'cache_hits', 'cache_accesses', 'cache_time', 'func_time', &
        ga_opts%ga_vars%name
    write (*, '(i5, 2i11, 3es16.8, 2i15, 2g11.2, 4es16.8)') &
        0, 0, 0, performance(1)%minn, performance(1)%meann, performance(1)%maxx, &
        cache%hits, cache%accesses, cache%time_cache, cache%time_func, &
        best_individual%cromosome
endif

! Cabeçalho do relatório para debugging                                 110
if (ga_opts%debug) then
    open(unit = debug_file, file = 'GA_debug', form = 'formatted')
    write (debug_file, '(a8, i3, a1, 1x, 5a16)') &
        '%[gen = ', gen, ']', 'fitness', ga_opts%ga_vars%name
    do i = 1, ga_opts%population_size
        write (debug_file, '(a13, 5es16.8)') '', fit(i), pop(i, :)
    enddo
    write (debug_file, *) ''
endif
                                                                         120
do gen = 1, ga_opts%ngenerations
    ! zera os contadores de eventos por geração
    ncrosses = 0
    nmutations = 0
    cache%hits = 0
    cache%accesses = 0

    ! Aplicação dos operadores genéticos
    pop = crossover(pop, fit)
    pop = mutate(pop)                                                    130

    ! Cálculo do fitness da população antes do elitismo
    fit = fitness(pop)

    ! Elitismo: substitui o indivíduo menos apto pelo mais apto da geração anterior
    max_fitness = maxval(fit)
    max_idx = maxloc(fit)
    if (max_fitness > best_individual%fitness) then
        ! Atualização da população e do fitness
        pop(max_idx(1), :) = best_individual%cromosome                   140
        fit(max_idx(1)) = best_individual%fitness
    endif

    ! Cálculo das estatísticas da nova geração
    best_individual%fitness = minval(fit)
    min_idx = minloc(fit)
    best_individual%cromosome = pop(min_idx(1), :)
    performance(gen + 1)%minn = best_individual%fitness
    performance(gen + 1)%meann = sum(fit)/ga_opts%population_size
    performance(gen + 1)%maxx = maxval(fit)                              150

    ! Imprime relatório de progresso (se ativada nas opções)
```

```
        if (ga_opts%print_progress) then
            write (*, '(i5, 2i11, 3e16.8, 2i15, 2g11.2, 4e16.8)') &
                  gen, ncrosses, nmutations, &
                  performance(gen + 1)%minn, performance(gen + 1)%meann, &
                  performance(gen + 1)%maxx, &
                  cache%hits, cache%accesses, cache%time_cache, cache%time_func, &
                  best_individual%cromosome
        endif                                                                    160

        ! Imprime toda a população e o valor de fitness (se a opção de debug
        ! estiver ativada)
        if (ga_opts%debug) then
            write (debug_file, '(a8, i3, a1, 1x, 5a16)') &
                  '%[gen = ', gen, ']', 'fitness', ga_opts%ga_vars%name
            do i = 1, ga_opts%population_size
                write (debug_file, '(a13, 5es16.8)') '', fit(i), pop(i, :)
            enddo
            write (debug_file, *) ''                                             170
        endif

        ! Contagem do número de gerações consecutivas sem melhoria
        if ( performance(gen + 1)%minn == performance(gen)%minn ) then
            nstall = nstall + 1
        else
            nstall = 0
        endif

        ! Critério de parada:                                                    180
        ! número máximo de gerações consecutivas sem melhoria atingido
        if (nstall == ga_opts%ng_stall) then
            print *, 'Critério de parada: Número máximo de gerações &
                      consecutivas sem melhoria atingido.'
            exit
        endif
    enddo

    call bestIndividual_GA(best_individual%cromosome)
end function run_ga                                                              190

function initialize_population() result (pop)
    implicit none
    real, dimension(ga_opts%population_size, ga_opts%num_vars) :: pop, rand_numbers, &
                                                                  min_value, range_value

    call random_number(rand_numbers)

    min_value = spread(ga_opts%ga_vars%min_value, dim = 1, ncopies = ga_opts%population_size)
    range_value = spread(ga_opts%ga_vars%max_value - ga_opts%ga_vars%min_value, dim = 1, &  200
                         ncopies = ga_opts%population_size)

    pop = min_value + rand_numbers*range_value
end function initialize_population

function crossover(pop, fit) result (children)
    implicit none
```

```
    real, dimension(:, :) :: pop
    real, dimension(:) :: fit                                                    210

    real, dimension(size(pop, dim=1), size(pop, dim=2)) :: parents, children

    integer :: couple
    real :: rand

    parents = roleta(pop, fit)
    children = parents

    do couple = 1, size(parents, dim=1) - 1, 2                                   220
        call random_number(rand)
        if (rand .le. ga_opts%pcrossover) then
            children(couple:couple + 1, :) = cross_pair(parents(couple, :), parents(couple + 1, :))
            ncrosses = ncrosses + 1
        endif
    enddo
end function crossover

function roleta(pop, fit) result (newpop)
    ! Implementa a seleção por roleta para um problema e que se deseja obter    230
    ! um mínimo.
    implicit none

    real, dimension(:, :) :: pop
    real, dimension(:) :: fit

    real, dimension(size(pop, dim=1), size(pop, dim=2)) :: newpop

    real :: max_fit, partsum, rand, slot
    integer :: indiv, i                                                          240

    ! Garante que todos os valores de fitness são positivos
    fit = fit + abs(minval(fit))

    ! Passagem do problema de mínimo para máximo
    max_fit = maxval(fit)
    if (max_fit /= 0.0) then
        fit = fit*(1.0/max_fit)
    else
        fit = fit + 1.0                                                          250
    endif
    fit = 2.0 - fit

    do indiv = 1, size(pop, dim=1)
        i = 0
        partsum = 0.0
        call random_number(rand)
        slot = rand*sum(fit)
        do
            if (partsum > slot) then                                             260
                exit
            endif
```

```
            i = i + 1
            partsum = partsum + fit(i)
        enddo
        newpop(indiv, :) = pop(i, :)
    enddo
end function roleta

function cross_pair(dad, mom) result (children)
    implicit none

    real, dimension(:) :: dad, mom

    real, dimension(2, size(dad)) :: children

    real :: rand
    integer :: crossVar

    children(1, :) = dad
    children(2, :) = mom

    ! Variável onde ocorrerá cruzamento
    call random_number(rand)
    crossVar = 1 + int(anint( rand*(size(dad) - 1) ))

    ! Cruzamento que permite filhos "entre" os pais
    call random_number(rand)
    children(:, crossVar) = rand*dad(crossVar) + (1.0 - rand)*mom(crossVar)
end function cross_pair

function mutate(pop) result (newpop)
    implicit none

    real, dimension(:, :) :: pop

    real, dimension(size(pop, dim=1), size(pop, dim=2)) :: newpop

    integer :: indiv, var
    real :: rand, vmin, vrange

    newpop = pop

    do indiv = 1, size(pop, dim=1)
        do var = 1, size(pop, dim=2)
            call random_number(rand)
            if (rand .le. ga_opts%pmutation) then
                vmin = ga_opts%ga_vars(var)%min_value
                vrange = ga_opts%ga_vars(var)%max_value - vmin
                call random_number(rand)
                newpop(indiv, var) = vmin + rand*vrange
                nmutations = nmutations + 1
            endif
        enddo
    enddo
end function mutate
```

```
    function fitness(pop) result (fit)
        implicit none
                                                                                    320
        real, dimension(:, :) :: pop

        real, dimension(size(pop, dim=1)) :: fit

        integer :: i, j
        logical :: cache_miss = .true.
        integer, dimension(1) :: min_hits
        real :: time_ref

        cache%time_cache = 0.0                                                      330
        cache%time_func = 0.0

        do i = 1, size(fit)
            time_ref = secnds(0.0)
            cache_miss = .true.
            do j = 1, cache%filled_position
                if ( all( (cache%cromosomes(j, :) == pop(i, :)) == .true. ) ) then
                    cache%accesses = cache%accesses + 1
                    cache%crom_hits(j) = cache%crom_hits(j) + 1
                    cache%hits = cache%hits + 1                                     340
                    fit(i) = cache%fitness(j)
                    cache_miss = .false.
                    exit
                else
                    cache%accesses = cache%accesses + 1
                endif
            enddo
            cache%time_cache = cache%time_cache + secnds(time_ref)
            time_ref = secnds(0.0)
            if (cache_miss) then                                                    350
                fit(i) = fitnessFunc(pop(i, :))

                if ( cache%filled_position < ga_opts%cache_size ) then
                    cache%filled_position = cache%filled_position + 1
                    min_hits(1) = cache%filled_position
                else
                    ! Cache replacement
                    min_hits = minloc(cache%crom_hits)
                endif
                cache%crom_hits(min_hits(1)) = 0                                    360
                cache%cromosomes(min_hits(1), :) = pop(i, :)
                cache%fitness(min_hits(1)) = fit(i)
            endif
            cache%time_func = cache%time_func + secnds(time_ref)
        enddo
    end function fitness
end module ga_mod
```