UNIVERSIDADE FEDERAL DE PERNAMBUCO
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Otimização Multiobjetivo de Confiabilidade e Custo via Algoritmos Evolucionários em Projeto de Sistemas

Isis Didier Lins

Orientador: Enrique Andrés López Droguett, PhD

Recife, 2007

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Otimização Multiobjetivo de Confiabilidade e Custo via Algoritmos Evolucionários em Projeto de Sistemas

Trabalho de Conclusão de Curso
por

Isis Didier Lins

Orientador: Enrique Andrés López Droguett, PhD

Recife, Agosto /2007

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Otimização Multiobjetivo de Confiabilidade e Custo via Algoritmos Evolucionários em Projeto de Sistemas

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia de Produção da Universidade Federal de Pernambuco - UFPE - como requisito parcial para obtenção de Grau em Engenharia de Produção.

Recife, Agosto /2007

L759o **Lins, Isis Didier**

Otimização multiobjetivo de confiabilidade e custo via algoritmos evolucionários em projeto de sistemas / Isis Didier Lins. – Recife: O Autor, 2007.

vii, 63 folhas. : il., fig.

Monografia (TCC) – Universidade Federal de Pernambuco. CTG. Engenharia de Produção, 2007.

Inclui bibliografia.

1. Engenharia de produção. 2. Pesquisa operacional. 3. Otimização multiobjetivo. 4. Projeto de sistemas – Algoritmos evolucionários. I. Título.

658.5 CDD (22.ed.) UFPE/B**CTG/2007-108**

AGRADECIMENTOS

Muitas pessoas contribuíram direta ou indiretamente para a realização deste Trabalho de Conclusão de Curso. Gostaria de agradecer em especial

- a meus pais Bernardete e Sóstenes e a meu irmão Lauro pelo constante apoio e incentivo incondicional.
- a Vicente pela parceria, compreensão e incentivo.
- ao Professor Enrique, pela dedicação durante o desenvolvimento deste trabalho.
- a minha turma: Sofia, Juju, Ayana, Luiz Meira, Yuri, Felipe, Luiz Jucá...
- a Paulo, Andrea, Márcio e todo o pessoal do RISCTEC.

RESUMO

Ao se projetar um sistema, deseja-se que ele seja altamente confiável e que os custos associados sejam os menores possíveis. No entanto, custo e confiabilidade, em geral, têm uma relação direta, ou seja, quanto maior a confiabilidade do sistema, maiores os seus custos. Isso torna os objetivos de se ter alta confiabilidade e baixos custos conflitantes. Assim, não é possivel obter, simultaneamente, desempenhos ótimos para ambos os objetivos e o que se encontra na prática é um conjunto de soluções ótimas segundo a abordagem multiobjetivo. A disposição física dos componentes do sistema interferem tanto na sua confiabilidade quanto nos seus custos. Um dos problemas de otimização relacionados a projetos de sistemas baseados em confiabilidade é o problema de alocação de redundâncias. Deseja-se encontrar o número ótimo de equipamentos para cada um dos subsistemas que formam o sistema. Esse problema pode ser visualizado como um problema de múltiplos objetivos, uma vez que o aumento do número de redundâncias incorre em maiores custos. Esse trabalho propõe uma metodologia multiobjetivo baseada em algoritmos genéticos para resolver o problema de alocação de redundâncias em confiabilidade. São considerados tanto sistemas não-reparáveis, que não são passíveis de manutenção, quanto sistemas reparáveis, que após a falha voltam à operação por um procedimento que não seja a sua completa substituição. São resolvidos dois exemplos de aplicação. O primeiro considera um sistema paralelo-série não-reparável em que se busca maximizar a confiabilidade e minimizar o custo de aquisição de componentes. O segundo leva em conta um sistema reparável paralelo-série, em que se quer maximizar a disponibilidade e minimizar os custos de aquisição e de manutenção corretiva dos componentes ao longo do tempo de missão estabelecido. Supõe-se que os componentes de tal sistema reparável têm processos de falha-reparo regidos por processos de renovação alternados. O processo de falha-reparo do sistema é resultante da superposição dos processos de falha-reparo dos componentes e não é caracterizado por um processo de renovação. Assim, a disponibilidade e o número de reparos dos componentes (utilizado no cálculo do custo) são obtidos por meio de simulação discreta de eventos.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

A confiabilidade pode ser entendida como a probabilidade de um item operar satisfatoriamente sob certas condições num período de tempo pré-determinado (RAUSAND; HOYLAND, 2003). Esse item pode ser desde um componente ou produto até um sistema complexo.

A confiabilidade de sistemas é um fator crítico em plantas industriais que apresentam alto risco para o meio ambiente e para a segurança dos indivíduos, como por exemplo, usinas nucleares, plataformas de petróleo e refinarias. Dessa maneira, a fase de projeto do sistema, isto é, o planejamento da disposição física dos subsistemas e seus respectivos componentes, deve levar em consideração não só as características inerentes ao processo produtivo, mas também a confiabilidade do sistema resultante e os custos associados.

Em geral, a relação entre confiabilidade do sistema e seu custo é direta: quanto maior a confiabilidade, maiores os custos. Logo, os objetivos de *projetar um sistema de alta confiabilidade* e *projetar um sistema com baixos custos* são conflitantes. Entretanto, é desejável que esses dois objetivos sejam considerados de maneira conjunta no momento do projeto do sistema, com o intuito de encontrar soluções satisfatórias para ambos.

A confiabilidade é uma métrica de desempenho de sistemas ou componentes *não-reparáveis*. Os sistemas não-reparáveis são descartados após a falha, isto é, eles não são passíveis de manutenção. Um sistema é dito *reparável* se ele é restaurado à operação por um procedimento que não seja a sua completa substituição. Se os tempos de reparo de tal sistema não são desprezíveis em relação ao tempo de operação, então a confiabilidade do mesmo é medida pela sua *disponibilidade*. Esta pode ser definida como a probabilidade de o sistema encontrar-se operacional em um dado instante de tempo (RAUSAND; HOYLAND, 2003). Assim como no caso da confiabilidade, os objetivos de *projetar um sistema de alta disponibilidade* e *projetar um sistema de baixos custos* normalmente são conflitantes.

Para se encontrar a melhor solução possível para um dado problema, recorre-se aos métodos de otimização, que normalmente consideram apenas um único objetivo (ver, por exemplo, (YALAOUI *et al.*, 2005)). No entanto, as situações práticas muitas vezes requerem que vários objetivos sejam atingidos – custo mínimo, máxima confiabilidade, mínimo risco, máxima disponibilidade, entre outros – e parte deles podem ser conflitantes. Assim, não há solução única que otimize simultaneamente todos os objetivos. O que se encontra, na prática, é um conjunto de soluções potenciais que são igualmente ótimas do ponto de vista multiobjetivo (COELLO *et al.*, 2002). Com base nesse conjunto, o decisor escolhe uma solução que apresente o balanceamento entre os objetivos de acordo com suas preferências.

Algoritmos de otimização baseados em computação evolucionária, tais como algoritmos genéticos, são de natureza probabilística e são usados principalmente quando o número de va-

riáveis é muito grande ou quando as funções-objetivo apresentam certas características não toleráveis pelos métodos clássicos de Programação Matemática, como por exemplo funções multimodais ou que não são deriváveis. Além disso, eles consideram várias soluções potenciais do problema simultaneamente e esta é uma característica bastante favorável para usar tais algoritmos na otimização de múltiplos objetivos (DEB, 1999).

No contexto da confiabilidade, redundâncias são, em linhas gerais, equipamentos adicionados ao sistema para aumentar a confiabilidade do mesmo. De acordo com Kuo *et al.* (2001), os problemas de otimização envolvendo projeto de sistemas baseados em confiabilidade podem ser classificados em:

- Problemas de alocação de redundâncias, em que as variáveis de decisão são o número de redundâncias.
- Problemas de alocação de confiabilidade, em que as variáveis de decisão são as confiabilidades dos componentes que constituem o sistema.
- Problemas de alocação de confiabilidade e redundâncias, em que as variáveis de decisão são a confiabilidade dos componentes e o número de redundâncias.
- Problemas de atribuição de componentes, quando o arranjo dos componentes no sistema fazem diferença na confiabilidade dos sistemas.

Esses problemas podem ser visualizados segundo uma abordagem multiobjetivo. O problema de alocação de redundâncias, por exemplo, pode envolver pelo menos dois objetivos, confiabilidade e custo, uma vez que o aumento do número de redundâncias no sistema não só aumenta a confiabilidade, mas também os custos associados. O problema de alocação de confiabilidade também pode considerar os objetivos de custo e confiabilidade, já que geralmente componentes mais confiáveis são mais caros. No caso de sistemas reparáveis, pode-se levar em conta o custo e a disponibilidade (em vez de confiabilidade).

Os problemas de otimização que envolvem projeto de sistemas e confiabilidade podem ser modelados e resolvidos por meio de algoritmos evolucionários, como é apresentado, por exemplo, em (BUSACCA *et al.*, 2001), (MARSEGUERRA *et al.*, 2005) e (TABOADA *et al.*, 2007). Nesse trabalho, será considerado o problema de alocação de redundâncias que será resolvido por meio de um método multiobjetivo baseado em algoritmos genéticos.

1.1 Justificativa

Muitas vezes as organizações consideram apenas o custo no momento do projeto dos sistemas, negligenciando a sua confiabilidade. Assim, apesar dos custos baixos, o funcionamento da planta pode não ser adequado, incorrendo em constantes paradas na produção devido à baixa

confiabilidade do sistema. Isso pode ser bastante oneroso para as empresas, pois a indisponibilidade do sistema incorre em custos que podem superar os custos que elas teriam ao manter um sistema mais confiável. Além disso, sistemas pouco confiáveis apresentam altos riscos à integridade dos indivíduos e de acidentes ambientais. Se esses riscos forem concretizados, a organização pode ter custos não só com indenizações e penas legais, mas também relacionados a sua imagem diante da sociedade. Para Kuo *et al.* (2001), a garantia de uma alta confiabilidade do sistema a preços competitivos é essencial para manter a competitividade entre as organizações.

Portanto, encontrar soluções potenciais que apresentem um balanceamento adequado entre confiabilidade e custo ainda na fase de projeto do sistema e que sejam ótimas no sentido multiobjetivo é pertinente. Dessa maneira, a planta pode operar com um nível de confiabilidade satisfatório e com custos que, apesar de não serem mínimos, são compensados por uma maior confiabilidade do sistema.

1.2 Objetivo Geral

O objetivo desse trabalho é desenvolver uma metodologia via algoritmos evolucionários para a otimização multiobjetivo da confiabilidade (disponibilidade, no caso de sistemas reparáveis) e custo durante o projeto de sistemas, levando em consideração o problema de alocação de redundâncias.

1.3 Objetivos Específicos

Para alcançar o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

- Entender a ligação da confiabilidade e as principais configurações de sistemas.
- Entender os conceitos relacionados a sistemas reparáveis.
- Fazer uma revisão bibliográfica dos principais métodos usados em otimização multiobjetivo dentro da área de confiabilidade, ressaltando as vantagens e desvantagens de cada um deles.
- Entender a estrutura geral de algoritmos evolucionários para um único objetivo e abordagens multiobjetivo existentes.
- Elaborar um modelo de otimização multiobjetivo de confiabilidade / disponibilidade e custo em projeto de sistemas via algoritmos genéticos.
- Aplicar a metodologia desenvolvida em dois exemplos.

1.4 Metodologia

Inicialmente, fez-se uma pesquisa bibliográfica sobre configurações de sistemas, otimização multiobjetivo em projeto de sistemas, métodos multiobjetivo tradicionais e algoritmos evolucionários multiobjetivo. Em seguida, implementou-se um algoritmo genético para resolver problemas de múltiplos objetivos em linguagem de programação C++, junto ao grupo de pesquisa RISCTEC-UFPE.

Foram considerados dois exemplos de aplicação que envolvem a alocação de redundâncias: o primeiro leva em conta os objetivos de confiabilidade e custo, no qual os componentes são não-reparáveis e possuem valores constantes de confiabilidade e o custo se refere ao custo de aquisição dos mesmos; o segundo é caracterizado pela otimização da disponibilidade e do custo de sistemas reparáveis com componentes sujeitos a manutenções corretivas.

Para resolver o segundo exemplo de aplicação, foi necessário o estudo de conceitos relacionados a sistemas reparáveis, bem como a simulação discreta de eventos. Na modelagem de tal exemplo considerou-se que:

- O sistema é formado de componentes dispostos em paralelo-série.
- Os componentes são reparáveis.
- Os tempos de reparo não são desprezíveis em relação aos tempos de operação.
- Cada componente é restaurado a uma condição de “tão bom quanto novo” após o reparo.
- Os processos de falha-reparo de cada componente são modelados por um processo de renovação alternado.
- O processo de falha-reparo do sistema paralelo-serie é o resultado de uma superposição de processos alternados. Ou seja, o processo de falha-reparo do sistema não necessariamente é caracterizado por um processo de renovação.

A disponibilidade de sistemas reparáveis constituídos de dois ou mais componentes dispostos em uma configuração diferente da disposição em série e sujeitos a reparos com durações *não* desprezíveis em relação ao tempo operacional não pode ser obtida por meio de uma expressão analítica. Dessa forma, optou-se por estimar a disponibilidade do sistema por meio de simulação discreta de eventos. Os custos consideram tanto o custo de aquisição das redundâncias quanto o custo das manutenções corretivas realizadas ao longo do tempo de missão.

Os resultados dos exemplos de aplicação foram obtidos por meio da ferramenta implementada durante a realização desse trabalho. Para o segundo exemplo, em conjunto com essa ferramenta, utilizou-se um simulador desenvolvido pelo RISCTEC-UFPE.

1.5 Organização do Trabalho

Além deste primeiro capítulo de introdução, esse trabalho possui mais cinco capítulos organizados da seguinte maneira:

- Capítulo 2: principais configurações de sistemas, diferenças entre sistemas reparáveis e não-reparáveis, conceitos de disponibilidade, descrição dos processos estocásticos relacionados à modelagem de componentes / sistemas reparáveis e introdução à simulação discreta de eventos.
- Capítulo 3: apresentação de conceitos relacionados à otimização multiobjetivo e dos principais métodos multiobjetivo tradicionais e descrição dos fundamentos de algoritmos evolucionários.
- Capítulo 4: desenvolvimento e descrição do algoritmo genético multiobjetivo.
- Capítulo 5: elaboração e resolução dos dois exemplos de aplicação.
- Capítulo 6: considerações finais.

2 CONFIABILIDADE E DISPONIBILIDADE DE SISTEMAS

A confiabilidade do sistema é geralmente expressa em termos da confiabilidade dos seus subsistemas ou componentes. Nesse trabalho, será adotada a seguinte terminologia (KUO *et al.*, 2001):

- Componente: é um conjunto de itens que representa um elemento auto-contido de um sistema.
- Subsistema: é um conjunto de componentes.
- Sistema: é formado por um conjunto de subsistemas.

De acordo com Kuo *et al.* (2001), para descrever a confiabilidade de um determinado sistema é preciso especificar o processo de falha dos equipamentos (componentes), a configuração do sistema e o estado em que o sistema é considerado falho.

O processo de falha do equipamento descreve a lei de probabilidade que governa as falhas. Considerando o tempo de vida de um comoponente como sendo a variável aleatória T, a probabilidade de ele falhar até um certo tempo t é dada pela distribuição acumulada:

$$F(t) = P[T \leq t], \quad t \geq 0. \tag{2.1}$$

A função confiabilidade do componente é definida como sendo a probabilidade de ele operar satisfatoriamente durante um período de tempo pré-determinado, isto é, ao longo do seu *tempo de missão*. Ela é matematicamente definida da seguinte forma:

$$R(t) = P[T > t] = 1 - P[T \leq t] = 1 - F_T(t), \quad t \geq 0. \tag{2.2}$$

Duas distribuições de probabilidade freqüentemente usadas para descrever o processo de falhas dos componentes são as distribuições exponencial (Equação (2.3)) e Weibull (Equação (2.4)).

$$f(t) = \begin{cases} \lambda e^{-\lambda t}, & t \geq 0, \lambda > 0 \\ 0, & \text{caso contrário.} \end{cases} \tag{2.3}$$

$$f_T(t) = \begin{cases} \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-\left(\frac{t}{\alpha}\right)^{\beta}}, & t > 0, \alpha > 0, \beta > 0 \\ 0, & \text{caso contrário.} \end{cases} \tag{2.4}$$

Diz-se que λ é o parâmetro da distribuição exponencial e que α e β são respectivamente os parâmetros de escala e de forma da distribuição de Weibull. As distribuições acumuladas e as funções confiabilidade para a distribuição exponencial são dadas pela Equações (2.5) e (2.6) e

para a distribuição de Weibull pelas Equações (2.7) e (2.8) nessa ordem.

$$F_T(t) = 1 - e^{-\lambda t}, \quad t \geq 0 \tag{2.5}$$

$$R_T(t) = e^{-\lambda t}, \quad t \geq 0 \tag{2.6}$$

$$F_T(t) = 1 - e^{-\left(\frac{t}{\alpha}\right)^{\beta}}, \quad t \geq 0 \tag{2.7}$$

$$R_T(t) = e^{-\left(\frac{t}{\alpha}\right)^{\beta}}, \quad t \geq 0 \tag{2.8}$$

A forma como os subsistemas e componentes estão fisicamente dispostos na planta – a configuração do sistema – considera as características produtivas do processo. O arranjo físico dos componentes no sistema pode ser representado por meio de um *diagrama de blocos*. Com o diagrama de blocos, é possível identificar quais componentes devem funcionar para que o sistema como um todo também funcione, ou seja, identificar os seus *caminhos mínimos* (MODARRES, 2006).

Para ilustrar, considere dois sistemas produtivos X e Y que produzem os produtos X e Y, respectivamente (Figura 2.1). A fabricação do produto X requer que a matéria-prima seja transformada pelos equipamentos 1 e 2, nessa ordem. Já a matéria-prima do produto Y pode ser processada pelo equipamento 3 ou, alternativamente, pelo equipamento 4. Diz-se que os equipamentos 1 e 2 estão em *série* e que os equipamentos 3 e 4 estão configurados em *paralelo*.

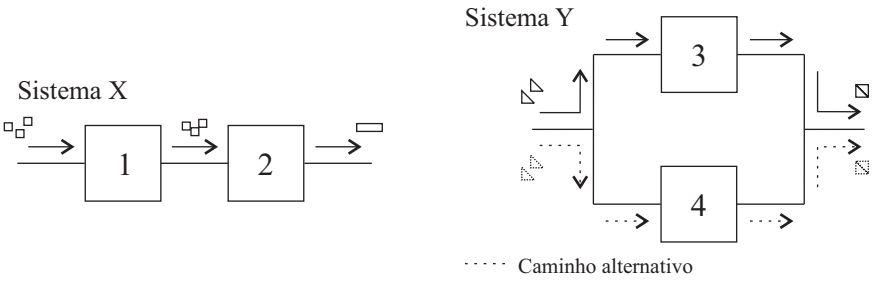

Figura 2.1: Exemplos de sistemas produtivos

Considerando ainda a Figura 2.1, os estados dos sistemas X e Y (operacional ou falho) podem ser avaliados por meio dos estados dos seus respectivos componentes. O sistema X está operacional se tanto o equipamento 1 quanto o equipamento 2 funcionarem, já que a falha de um dos dois equipamentos impede que o produto X chegue a seu estado final. Já o sistema Y, em caso de falha do equipamento 3, possui um caminho alternativo (equipamento 4) para fabricar o produto Y e só é considerado falho se os componentes 3 e 4 falharam simultaneamente. A Tabela 2.1 e a Tabela 2.2 mostram os possíveis estados dos sistemas X e Y em função dos estados dos seus respectivos componentes.

Tabela 2.1: Estados do sistema X em função dos estados dos componentes 1 e 2

Componente 1	**Componente 2**	**Sistema X**
Operacional	Operacional	Operacional
Operacional	Falho	Falho
Falho	Operacional	Falho
Falho	Falho	Falho

Tabela 2.2: Estados do sistema Y em função dos estados dos componentes 3 e 4

Componente 3	**Componente 4**	**Sistema Y**
Operacional	Operacional	Operacional
Operacional	Falho	Operacional
Falho	Operacional	Operacional
Falho	Falho	Falho

2.1 Configurações de Sistemas

A seguir serão mostradas os principais tipos de configurações de sistemas. Considere um sistema formado por n componentes, onde $P[O_i]$, $i = 1,2,\ldots,n$ é a probabilidade de ocorrência do evento O_i de que o componente i opera satisfatoriamente por um período de tempo pré-determinado ($P[O_i] = r_i$ é a confiabilidade do componente i) e que $\overline{O_i}$ é o evento complementar de O_i. Assume-se ainda que os componentes operam de maneira independente e que o sistema pode estar operacional ou falho. Para simplificação das análises, os valores para as confiabilidades dos componentes são considerados constantes, no entanto, esses valores poderiam ser obtidos da função confiabilidade caso as distribuições de probabilidade de falhas dos componentes fossem conhecidas.

2.1.1 Configuração em Série

Em um sistema em série, como ilustrado na Figura 2.2, os n componentes devem estar operacionais para que o sistema esteja operacional. Basta um dos n componentes falhar para que o sistema passe para o estado falho. A configuração em série é a mais simples que um sistema pode ter e é bastante comum em sistemas produtivos.

A confiabilidade do sistema em série é calculada da seguinte forma:

$$R_S = P[O_1 \cap O_2 \cap \cdots \cap O_n] = \prod_{i=1}^{n} P[O_i]$$

$$R_S = \prod_{i=1}^{n} r_i \tag{2.9}$$

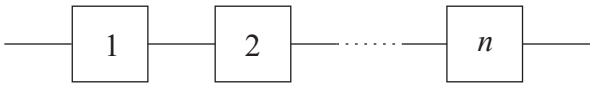

Figura 2.2: Sistema em série

2.1.2 Configuração em Paralelo

A Figura 2.3 mostra um sistema em paralelo. O sistema é considerado operacional se qualquer um de seus componentes se encontrar operacional e só falha quando todos os seus componentes falham. A confiabilidade do sistema em paralelo é dada pela seguinte expressão:

$$R_S = P[O_1 \cup O_2 \cup \cdots \cup O_n] = 1 - P[\overline{O}_1 \cap \overline{O}_2 \cap \cdots \cap \overline{O}_n] = 1 - \prod_{i=1}^{n} P[\overline{O}_i] = 1 - \prod_{i=1}^{n} (1 - P[O_i])$$

$$R_S = 1 - \prod_{i=1}^{n} (1 - r_i) \tag{2.10}$$

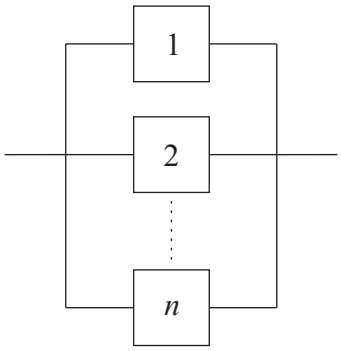

Figura 2.3: Sistema em paralelo

A configuração em paralelo também é conhecida como configuração redundante. No entanto, o termo *redundante* só deve ser usado quando a configuração do sistema é modificada para produzir caminhos adicionais a fim de aumentar a confiabilidade do sistema. Dessa maneira, um sistema paralelo pode resultar da estrutura básica do sistema, em que existem vários caminhos executando as mesmas operações, ou pode ser obtido por meio do uso de redundâncias num sistema cujo projeto ou reprojeto é baseado em confiabilidade (KUO *et al.*, 2001). O sistema Y da Figura 2.1 possui uma redundância (componente 4).

Quanto maior o número de fatores no produtório da equação 2.10, maior será o valor da confiabilidade do sistema R_S. Entretanto, incrementar o número de redundâncias aumenta os custos relacionados com a aquisição e a manutenção das mesmas. Portanto, é interessante encontrar o número de componentes que maximize a confiabilidade do sistema, mas que também minimize os custos associados.

2.1.3 Configuração Série-Paralelo

Seja um sistema formado por k subsistemas conectados em paralelo. Cada subsistema i é composto de n_i, $i = 1,2,\ldots,k$, componentes em série. A Figura 2.4 ilustra um sistema série-paralelo. Seja R_i a confiabilidade do subsistema i e r_{ij} a confiabilidade do componente j, $j = 1,2,\ldots n_i$ do subsistema i. Assim,

$$R_i = \prod_{j=1}^{n_i} r_{ij} \tag{2.11}$$

$$R_S = 1 - \prod_{i=1}^{k}(1 - R_i). \tag{2.12}$$

Juntando as expressões 2.11 e 2.12, tem-se que

$$R_S = 1 - \prod_{i=1}^{k}(1 - \prod_{j=1}^{n_i} r_{ij}). \tag{2.13}$$

Se os componentes em cada subsistema i forem idênticos, a equação 2.13 pode ser reduzida para

$$R_S = 1 - \prod_{i=1}^{k}(1 - r_i^{n_i}), \tag{2.14}$$

em que r_i é a confiabilidade de cada componente do subsistema i, $i = 1,2,\cdots,k$.

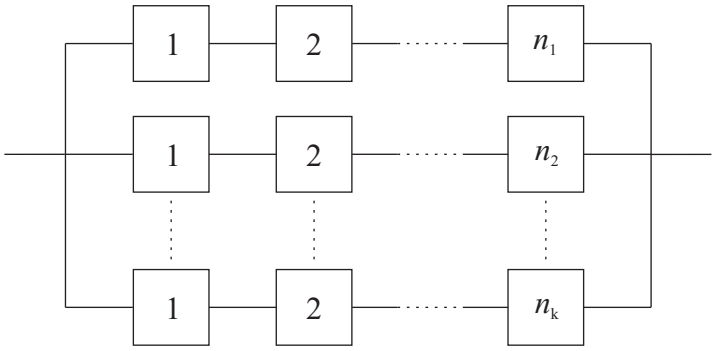

Figura 2.4: Sistema em série-paralelo

2.1.4 Configuração em Paralelo-Série

Seja um sistema composto por k subsistemas em paralelo. Cada subsistema i é formado por n_i, $i = 1,2,\ldots,k$, componentes em paralelo. A Figura 2.5 ilustra uma configuração paralelo-

série. Se R_i é a confiabilidade do sistema i e r_{ij} é a confiabilidade do componente j, $j = 1,2,\ldots,n_i$ do subsistema i, então:

$$R_i = 1 - \prod_{j=1}^{n_i}(1 - r_{ij}) \tag{2.15}$$

$$R_S = \prod_{i=1}^{k} R_i. \tag{2.16}$$

Assim, a confiabilidade do sistema é dada por

$$R_S = \prod_{i=1}^{k}\left[1 - \prod_{j=1}^{n_i}(1 - r_{ij})\right]. \tag{2.17}$$

Se todos os componentes de cada subsistema i forem idênticos, então a expressão 2.17 pode ser reduzida para

$$R_S = \prod_{i=1}^{k}\left[1 - (1 - r_i)^{n_i}\right], \tag{2.18}$$

onde r_i é a confiabilidade dos componentes do subsistema i.

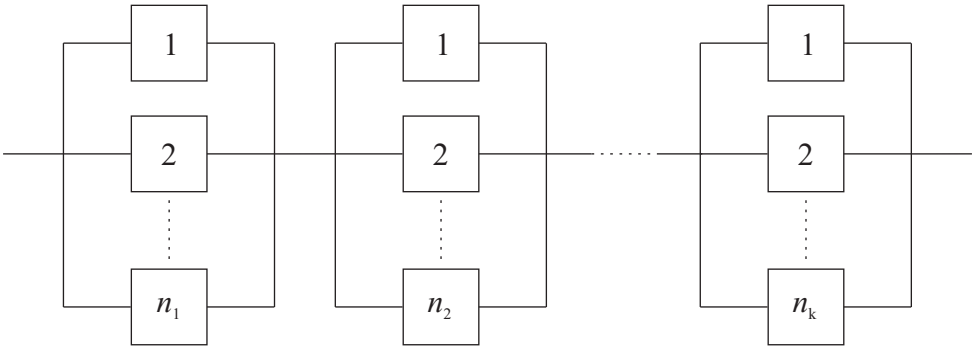

Figura 2.5: Sistema em paralelo-série

2.1.5 Sistemas Série-Paralelo Hierárquicos

Em geral, os sistemas não são configurados em série ou em paralelo, mas apresentam uma combinação das duas configurações (MODARRES, 2006). Um sistema é chamado de *sistema série-paralelo hierárquico* se ele pode ser visto como um conjunto de subsistemas dispostos em série ou paralelo. O sistema série-paralelo hierárquico da Figura 2.6 possui dois subsistemas em série: {1, 2, 3, 4, 5} (subsistema A) e {6, 7} (subsistema B). O subsistema A é, por sua vez,

formado pelos subsistemas C e D ({1, 2, 3} e {4, 5} respectivamente). O subsistema C possui o componente 1 em série com o subsistema que tem os componentes 2 e 3 em paralelo. Já o subsistema D é formado pelos componentes 4 e 5 em paralelo. O subsistema B é constituído pelos componentes 6 e 7 em paralelo. Seja R_A, R_B, R_C, R_D as confiabilidades dos subsistemas A, B, C e D nessa ordem. Então

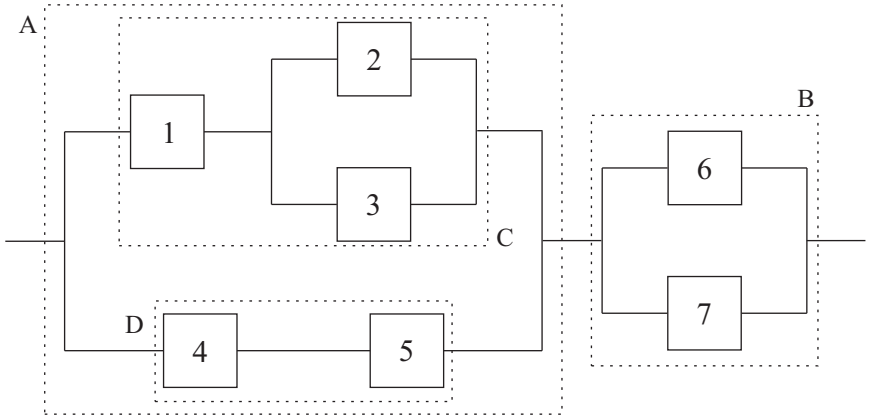

Figura 2.6: Exemplo de sistema série-paralelo hierárquico

$$R_A = 1-(1-R_C)\cdot(1-R_D) \tag{2.19}$$

$$R_B = 1-(1-r_6)\cdot(1-r_7) \tag{2.20}$$

$$R_C = r_1\cdot[1-(1-r_2)\cdot(1-r_3)] \tag{2.21}$$

$$R_D = r_4\cdot r_5, \tag{2.22}$$

em que r_i, $i = 1,2,3,4,5,6$ e 7 representa a confiabilidade do i-ésimo componente. A confiabilidade do sistema R_S é dada combinando-se as expressões (2.19)-(2.22), uma vez que $R_S = R_A \cdot R_B$:

$$R_S = \big\{1-\{1-r_1\cdot[1-(1-r_2)(1-r_3)]\}\cdot(1-r_4\cdot r_5)\big\}\cdot\big\{1-(1-r_6)(1-r_7)\big\} \tag{2.23}$$

É importante notar que o cálculo da confiabilidade do sistema depende da configuração que ele apresenta. Em geral, sistemas paralelo-série têm maior confiabilidade que sistemas configurados em série-paralelo, se o mesmo conjunto de componentes é usado em ambos os sistemas. No entanto, os custos associados às redundâncias também devem ser levados em consideração.

Existem outros tipos de configuração, como por exemplo as configurações complexas. Para mais detalhes em configuração de sistemas, consultar (KUO *et al.*, 2001), (MODARRES, 2006) e (RAUSAND; HOYLAND, 2003).

2.2 Sistemas Não-Reparáveis

Diz-se que um sistema é não-reparável se, após a sua falha – a primeira e única –, for necessária a sua completa substituição para que ele retorne à condição operacional. Ou seja, ele não é passível de manutenção.

O tempo de vida de um sistema não-reparável é caracterizado pela variável aleatória T. Como a falha de um sistema não afeta o desempenho de outro localizado em outro lugar, assume-se que os tempos de vida dos diferentes sistemas são independentes. Além disso, se cópias de um sistema são produzidas por um mesmo fabricante, assume-se que todas possuem a mesma ditribuição para o tempo de vida. Combinando-se essas duas hipóteses, tem-se que os tempos de vida são independentes e identicamente distribuídos (RIGDON; BASU, 2000).

A distribuição acumulada $F(t)$ do tempo de vida de componentes não-reparáveis é dada pela equação 2.1 e a função confiabilidade $R(t)$ é apresentada na equação 2.2. A função de densidade de probabilidade é definida como sendo a derivada da função de distribuição acumulada:

$$f(t) = \frac{dF(t)}{dt}, \quad t \geq 0. \tag{2.24}$$

As equações (2.3) e (2.4) são exemplos de densidade de probabilidade. Uma outra medida relacionada aos componentes não-reparáveis é a taxa de falha $h(t)$. Ela é o limite da probabilidade de uma unidade falhar, pela primeira e única vez, em um pequeno intervalo de tempo dado que ela funcionou desde o início do intervalo de observação:

$$h(t) = \lim_{\Delta t \to 0} \frac{P[t < T \leq t + \Delta t | T > t]}{\Delta t}, \quad t \geq 0. \tag{2.25}$$

Para ser uma função de densidade de probabilidade, é preciso que a sua integral em todo o domínio da variável considerada (no caso, a variável t) seja igual à unidade. Apesar de ser definida como o limite de uma probabilidade condicional, a taxa de falha não é uma probabilidade condicional, uma vez que não necessariamente apresenta essa característica, ou seja, em geral,

$$\int_0^\infty h(\tau) d\tau \neq 0. \tag{2.26}$$

Existem três fases características para componentes não-reparáveis: a fase de *mortalidade infantil* ou *burn-in*, na qual a taxa de falha do sistema é decrescente ($\frac{dh(t)}{dt} < 0$); a fase de *vida útil*, em que a taxa de falha é praticamente constante; a fase de desgaste ou degradação, quando a taxa de falha é crescente ($\frac{dh(t)}{dt} > 0$). Essas três fases caracterizam a curva da banheira mostrada na figura 2.7.

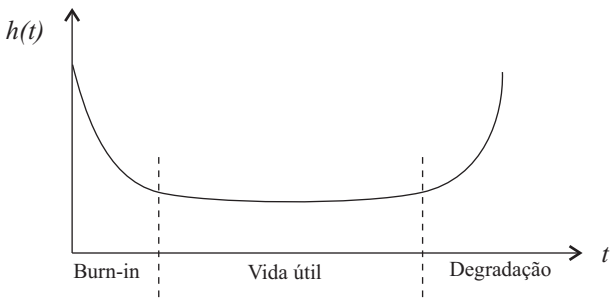

Figura 2.7: Curva da banheira

2.3 Sistemas Reparáveis

Ao ocorrer uma falha em um sistema, diz-se que ele é reparável se ele pode ser restituíduo à operação por algum processo de reparo que não seja a sua completa substituição (RIGDON; BASU, 2000).

Algumas definições relacionadas aos sistemas reparáveis são dadas a seguir:

- Tempo global: é o tempo observado desde que o sistema foi colocado em operação pela primeira vez. Em um sistema reparável, observam-se vários tempos de falha $0 < T_1 < T_2 < \cdots$ medidos com base no tempo global. Sistemas reparáveis podem ser analisados como sistemas não-reparáveis até a primeira falha.

- Tempo local: é o tempo decorrido entre duas falhas consecutivas. Os tempos entre falhas são definidos como sendo a diferença entre dois tempos globais consecutivos, ou seja,

$$\begin{aligned} X_1 &= T_1 - T_0, \quad T_0 = 0 \\ X_2 &= T_2 - T_1 \\ X_3 &= T_3 - T_2 \\ &\vdots \end{aligned} \tag{2.27}$$

- Deterioração e melhoria: diz-se que um sistema reparável sofre *deterioração* se os tempos entre falhas tendem a ser menores com o passar do tempo (Figura 2.8). Caso os tempos entre falhas consecutivas sejam cada vez mais espaçados, diz-se que o sistema reparável está em processo de *melhoria* (Figura 2.9).

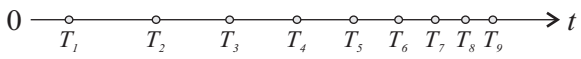

Figura 2.8: Processo de deterioração

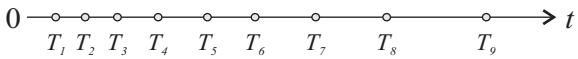

Figura 2.9: Processo de melhoria

Os tempos de falha observados não necessariamente são medidos de acordo com o "tempo de relógio" e dependem do tipo e das características dos sistemas e componentes analisados. Por exemplo, para medir a "idade" de automóveis, a quilometragem é uma métrica mais apropriadada do que o tempo de uso, enquanto que, para avaliar a "idade" de um avião, indica-se o número de horas de operação (RIGDON; BASU, 2000).

O reparo – ou *manutenção corretiva* – é executado após a falha do componente e tem como objetivo retorná-lo a um estado operacional o mais breve possível (RAUSAND; HOYLAND, 2003).

Uma métrica relacionada aos sistemas reparáveis é a *disponibilidade*.

2.3.1 Disponibilidade

Há quatro métricas de disponibilidade (RAUSAND; HOYLAND, 2003):

- Disponibilidade instantânea.
- Disponibilidade limite.
- Disponibilidade média.
- Disponibilidade média limite.

A disponbilidade instantânea é definida pela expressão

$$A(t) = P[X(t) = 1] = E[X(t)], \quad t \geq 0, \tag{2.28}$$

onde $E[X(t)]$ é o valor esperado da variável aleatória $X(t)$ dado pela equação a seguir:

$$E[X(t)] = 0 \cdot P[X(t) = 0] + 1 \cdot P[X(t) = 1] = P[X(t) = 1]. \tag{2.29}$$

Sob certas condições, para determinar a disponibilidade $A(t)$, é freqüentemente indicado o uso de processos de renovação (RAUSAND; HOYLAND, 2003), que são um tipo de processos de contagem. Nas próximas seções será dada uma introdução sobre processos de contagem, e mais especificamente sobre processos de renovação.

A disponibilidade limite é dada por

$$A = \lim_{t \to \infty} A(t), \tag{2.30}$$

quando o limite existe. Ela pode ser interpretada como a fração do tempo em que o item está disponível considerando um longo período.

A disponibilidade média no intervalo $(0,t]$ é definida por:

$$A_m(0,t) = \frac{1}{t}\int_0^t A(\tau)\,d\tau, \tag{2.31}$$

Ela indica a proporção esperada de $(0,t]$ em que o sistema encontra-se operacional.

A disponibilidade média limite no intervalo $(0,t]$ é apresentada a seguir:

$$A_{m,\infty} = \lim_{t\to\infty} A_m(0,t). \tag{2.32}$$

Se a disponibilidade limite existe, então a disponibilidade média limite é igual à disponibilidade limite.

As indisponibilidades instantânea, limite, média e média limite são o complemento das disponibilidades instantânea, limite, média e média limite nessa ordem e são representadas por U seguido dos respectivos índices de cada métrica. A indisponibilidade instantânea, por exemplo, é dada por:

$$U(t) = 1 - A(t) = 1 - P[X(t) = 1] = P[X(t) = 0], \quad t \geq 0. \tag{2.33}$$

2.3.1.1 Disponibilidade de Sistemas

Considere um sistema S formado por n componentes que operam de maneira independente. As variáveis $X_i(t)$, $i = 1,2,\ldots,n$ indicam o estado do i-ésimo componente no instante t. Os valores A_i, $i = 1,2,\ldots,n$ representam a disponibilidade média do componente i. A disponibilidade média do sistema A_S é dada conforme a lógica que ele apresenta, isto é, como os componentes se relacionam no sistema. Dessa maneira, em vez de considerar as confiabilidades dos componentes nas expressões (2.9), (2.10), (2.13), (2.17) e (2.23) apresentadas na seção (2.1), consideram-se as disponibilidades médias dos componentes. Obtêm-se então as disponibilidades médias dos sistemas em série, paralelo, série-paralelo, paralelo-série e série-paralelo hierárquico (figura 2.6) nessa ordem.

De acordo com Rausand e Hoyland (2003), para usar a abordagem de diagrama de blocos no cálculo da disponibilidade de sistemas, assume-se que os comoponentes falham e são reparados independentemente uns dos outros. Isso significa que a operação e a manutenção de um componente não é influenciada pelo estado dos outros componentes, o que nem sempre acontece na prática. Além disso, medidas de desempenho dinâmicas, tais como a disponibilidade instantânea do sistema considerado, não podem ser obtidas dessa forma.

A simulação discreta de eventos (seção 2.5) é uma alternativa para avaliar o desempenho

do sistema dinamicamente. Pode-se obter, por exemplo, a disponibilidade instantânea e a disponibiliade média ao longo do período de simulação. Essa última pode ser dada pela fração do tempo de simulação em que o sistema esteve operacional:

$$A_{m,S} = \frac{\text{Tempo operacional}}{\text{Tempo total}}. \tag{2.34}$$

2.4 Processos de Contagem

A variável aleatória $N(t)$ denota o número de eventos que ocorreram no intervalo $[0,t]$. $N(t)$ é, portanto, uma variável de contagem. No contexto de sistemas reparáveis, os eventos são basicamente as falhas do sistema. Considera-se que ao falhar, o sistema é reparado a um estado operacional e que o tempo de reparo é desprezível em relação ao tempo de operação.

Em geral, os tempos entre falhas definidos em (2.27) não são independentes ou identicamente distribuídos, a menos que o sistema seja reparado após a falha a uma condição de "tão bom quanto novo" e as condições ambientais e operacionais permaneçam constantes durante todo o período de missão (RAUSAND; HOYLAND, 2003).

Diz-se que o processo estocástico $\{N(t), t \geq 0\}$ é um processo de contagem se ele satisfaz (ROSS, 2000):

- $N(t) \geq 0$.
- $N(t)$ assume valores inteiros.
- Se $s < t$ então $N(s) \leq N(t)$.
- Para $s < t$,$N(t) - N(s)$ representa o número de eventos ocorridos no intervalo $(s,t]$.

Algumas definições são necessárias quando se consideram processos de contagem (RIGDON; BASU, 2000; RAUSAND; HOYLAND, 2003):

- Incrementos independentes: um processo pontual tem *incrementos independentes* se, para todo n e para todo $s_1 < t_1 \leq s_2 < t_2 \leq \cdots < s_n \leq t_n$, as variáveis aleatórias $N(s_1,t_1]$, $N(s_2,t_2]$, ..., $N(s_n,t_n]$ são independentes, ou seja,

  $$P\left[N(s_1,t_1] = k_1, N(s_2,t_2] = k_2, \ldots, N(s_n,t_n] = k_n\right] = \prod_{i=1}^{n} P[N(s_i,t_i] = k_i]. \tag{2.35}$$

  O número de falhas em um intervalo não é influenciado pelo número de falhas ocorridas em um intervalo imediatamente anterior (sem superposição dos intervalos).

- Incrementos estacionários: diz-se que um processo de contagem tem *incrementos estacionários* se $P[N(t,t+s] = k]]$ é independente de t para todo k. Isso significa que a distribui-

ção do número de falhas em um intervalo de tempo depende apenas do seu comprimento e não da distância do intervalo à origem.

- Processo regular: um processo de contagem é *regular* se

$$P[N(t+\Delta t) - N(t) \geq 2] = o(\Delta t), \tag{2.36}$$

quando Δt é pequeno e $o(\Delta t)$ é uma função de Δt com a propriedade de que $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$. Isso denota, na prática que não há falhas simultâneas no sistema.

- ROCOF: a taxa de ocorrência de falhas (*Rate of Occurrence of Failures* - ROCOF) de um processo de contagem é definida como

$$w(t) = \frac{dW(t)}{dt} = \frac{dE[N(t)]}{dt}, \quad t \geq 0, \tag{2.37}$$

onde $W(t) = E[N(t)]$ e $E[N(t)]$ é o número esperado de falhas no intervalo $(0,t]$. A ROCOF pode ser interpretada como a taxa instantânea de mudança no número esperado de falhas.

- A *função intensidade* $\lambda(t)$ de um processo de contagem é dada por

$$\lambda(t) = \lim_{\Delta t \to 0} = \frac{P[N(t,t+\Delta t] \geq 1]}{\Delta t}. \tag{2.38}$$

Diferentemente da taxa de falha (Equação (2.25)), a função intensidade de falha é a probabilidade incondicional de que uma falha – não necessariamente a primeira – ocorra em um pequeno intervalo, dividida pelo comprimento desse intervalo. A ROCOF (equação (2.37)) e a função intensidade de falhas são iguais, se falhas simultâneas não forem permitidas (RIGDON; BASU, 2000) e ambas são métricas de confiabilidade de sistemas não-reparáveis.

Os três principais processos de contagem usados em confiabilidade são:

- Processo Homogêneo de Poisson (HPP - *Homogeneous Poisson Process*).
- Processo de Renovação.
- Processo Não-Homogêneo de Poisson (NHPP - *Nonhomogeneous Poisson Process*).

Em um HPP, os tempos entre falhas (expressões em (2.27)) são independentes e *exponencialmente* distribuídos com o mesmo parâmetro λ (taxa de falha).

O NHPP difere do HPP apenas no fato de que a taxa de ocorrência de falhas varia com o tempo em vez de ser constante. Assim, no NHPP, os tempos entre falhas não são nem independentes nem identicamente distribuídos. Segundo (RAUSAND; HOYLAND, 2003), esse processo estocástico é usualmente utilizado para modelar sistemas reparáveis em que o componente ao falhar sofre um *reparo mínimo*, isto é, o componente volta à operação com a mesma condição que apresentava imediadamente antes da ocorrência da falha ("tão ruim quanto antes").

O processo de renovação é um processo de contagem em que os tempos entre falhas são independentes e identicamente distribuídos com uma distribuição arbitrária. Após a falha, o componente é submetido a um *reparo perfeito*, isto é, o componente é restituído à mesma condição que apresentava quando novo ("tão bom quanto novo").

Tanto o NHPP quanto o processo de renovação são generalizações para o HPP. Além disso, o NHPP e o processo de renovação representam dois tipos extremos de reparo: o reparo mínimo e o reparo perfeito, respectivamente. Entretanto, as ações de reparo encontram-se normalmente entre esses dois extremos e diz-se que os reparos são *imperfeitos*. Processos de Renovação Generalizados (GRP - *Generalized Renewal Processes*), por exemplo, podem ser usados para modelar sistemas reparáveis submetidos a reparos imperfeitos (YAÑES *et al.*, 2002).

A seguir dá-se uma introdução sobre processos de renovação. Para maiores detalhes em HPP e NHPP recomenda-se (ROSS, 2000), (RAUSAND; HOYLAND, 2003) e (RIGDON; BASU, 2000).

2.4.1 Processos de Renovação

De acordo com Rigdon e Basu (2000), diz-se que um processo de contagem é um processo de renovação se os tempos entre eventos (falhas de um componente, por exemplo) $X_1, X_2, \ldots$ são independentes e identicamente distribuídos. Uma suposição desse tipo de modelagem é a de que a ação de reparo é perfeita e restitui o componente à condição de "tão bom quanto novo" após a ocorrência de uma falha. Isso equivale a substituir o equipamento falho por um outro com as mesmas características operacionais. Assim, processos de renovação não podem ser usados para modelar componentes que sofrem deterioração. Assume-se ainda que os tempos de reparo são desprezíveis em relação ao tempo em que o componente permanece em operação. Os eventos observados são chamados *renovações*.

Os tempos entre falhas $X_1, X_2, \ldots$ tem função de distribuição

$$F_X(x) = P[X_i < x], \quad x > 0, \quad i = 1, 2, \ldots. \tag{2.39}$$

A função de distribuição $F_X(x)$ é a *distribuição subjacente do processo de renovação*. Em alguns casos, X_1 tem função de distribuição $F_{X_1}(x)$ diferente da função de distribuição dos demais tempos entre falhas $F_X(x)$. O processo que apresenta essa característica é chamado

de *processo de renovação modificado*.

O tempo até o r-ésimo evento T_r é dado por

$$T_r = X_1 + X_2 + \cdots + X_r = \sum_{i=1}^{r} X_i. \tag{2.40}$$

Para estabelecer a distribuição de T_r, precisa-se do conceito de *convolução* (RAUSAND; HOYLAND, 2003). Sejam X_1 e X_2 tempos entre falhas com funções de distribuição contínuas F_1 e F_2 e respectivas densidades f_1 e f_2. Fazendo $X = X_1 + X_2$, a função de distribuição de X é chamada de convolução de F_1 e F_2,

$$F_X(x) = \int_0^x F_1(x-\tau)\,dF_2(\tau) = \int_0^x F_2(x-\tau)\,dF_1(\tau). \tag{2.41}$$

Se F_1 e F_2 são tais que $\frac{dF_X(x)}{dx}$ pode ser obtida por meio da diferenciação dentro da integral, então a função densidade de X é chamada de convolução de f_1 e f_2,

$$f_X(x) = \int_0^x f_1(x-\tau)\,f_2(\tau)\,d\tau = \int_0^x f_2(x-\tau)\,f_1(\tau)\,d\tau. \tag{2.42}$$

Uma vez que T_{r-1} e X_r são independentes e que $T_r = T_{r-1} + X_r$, a função de distribuição $F^{(r)}(t)$ é a convolução das distribuições de T_{r-1} e X_r:

$$F^{(r)}(t) = \int_0^t F^{(r-1)}(t-\tau)\,dF_X(\tau). \tag{2.43}$$

Quando os períodos de renovação têm distribuição contínua com função densidade de probabilidade $f_X(x)$, então a função densidade de probabilidade de T_r é dada por:

$$f^{(r)}(t) = \int_0^t f^{(r-1)}(t-\tau)\,df_X(\tau). \tag{2.44}$$

Pela integração sucessiva de (2.44) para $r = 2,3,4,\ldots$, a função densidade de probabilidade de T_r para um r específico pode, em princípio, ser encontrada:

$$\begin{aligned} f^{(2)}(t) &= \textstyle\int_0^t f^{(1)}(t-\tau)\,df_X(\tau), \quad \text{onde } f^{(1)}(t) = f_{X_1}(t) \\ &\vdots \\ f^{(r)}(t) &= \textstyle\int_0^t f^{(r-1)}(t-\tau)\,df_X(\tau) \end{aligned} \tag{2.45}$$

A distruibuição de T_r também pode ser obtida usando-se transformadas de Laplace (RAUSAND; HOYLAND, 2003).

O número de renovações no intervalo $(0,t]$ é

$$N(t) = \max\{r, T_r \leq t\}. \tag{2.46}$$

Pela definição de $N(t)$, tem-se que o número de reparos em $(0,t]$ é maior ou igual a r se e somente se r renovações tenham ocorrido até o tempo t, isto é,

$$N(t) \geq r \Leftrightarrow T_r \leq t \quad \Rightarrow \quad P[N(t) \geq r] = P[T_r(t) \leq r].$$

Então,

$$P[N(t) = r] = P[N(t) \geq r] - P[N(t) \geq r+1] = F^{(r)}(t) - F^{(r+1)}(t). \tag{2.47}$$

A *função de renovação* $W(t)$ é o número esperado de renovações no intervalo $(0,t]$, isto é,

$$W(t) = E[N(t)]. \tag{2.48}$$

Ela também pode ser expressa como

$$W(t) = \sum_{r=1}^{\infty} F^{(r)}(t) \tag{2.49}$$

ou pela equação integral, se forem combinadas as expressões 2.43 e 2.49,

$$W(t) = F_{X_1}(t) + \int_0^t W(t-\tau)\, dF_X(\tau). \tag{2.50}$$

A equação 2.50 é a *equação de renovação fundamental* e algumas vezes pode ser resolvida para $W(t)$.

A densidade de renovação é simplesmente a derivada da função de renovação e coincide com a ROCOF definida em (2.37):

$$w(t) = \frac{dW(t)}{dt}. \tag{2.51}$$

2.4.2 Processos de Renovação Alternados

Suponha que um componente é posto em operação no tempo $t = 0$. Toda vez que falha, ele é substituído por um componente idêntico ou é reparado à condição de "tão bom quanto novo". Assume-se que os tempos entre falhas $X_1, X_2, \ldots$ são independentes e identicamente distribuídos com função de distribuição acumulada $F(x) = P[X_i \leq x], i = 1,2,\ldots$ e tempo médio entre falhas MTBF (*mean time between failures*). Após a falha, ele fica um período de tempo indisponível devido à ação de reparo. Os tempos de reparo $D_1, D_2, \ldots$ são considerados independentes e identicamente distribuídos com função de distribuição acumulada $F(d) = [D_i \leq d], i = 1,2,\ldots$

e tempo médio de reparo MTTR (*mean time to repair*). Considera-se ainda que $X_i + D_i$, $i = 1,2,\ldots$ são independentes.

O estado do componente é dado pela variável binária:

$$X(t) = \begin{cases} 1, & \text{se o comoponente está operacional no tempo } t \\ 0, & \text{caso contrário.} \end{cases} \tag{2.52}$$

e a figura 2.10 ilustra a seqüência de falhas e reparos ao longo do tempo.

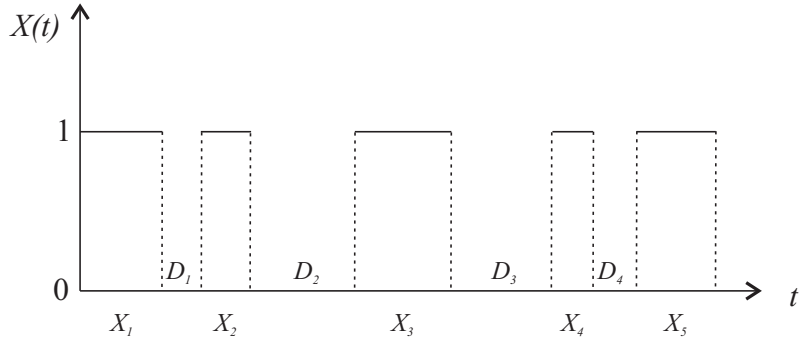

Figura 2.10: Processo de renovação alternado

Um processo de falha-reparo como o ilustrado na figura 2.10 é chamado de *processo de renovação alternado*. Se os eventos considerados indicarem o término de um reparo, então os tempos entre eventos são $Y_i = X_i + D_i$, $i = 1,2,\ldots$ e se tem um processo de renovação com a função de distriuição subjacente $H(y)$ que é a convolução de $F_X(x)$ e $F_D(d)$ (funções de distribuição dos tempos entre falhas e dos tempos entre reparos, respectivamente):

$$H(y) = P[Y_i \leq y] = P[X_i + D_i \leq y] = \int_0^y F_X(y-\tau)\, dF_D(\tau). \tag{2.53}$$

Se os eventos forem as ocorrências de falha, então se trata de um processo de renovação modificado com $Y_1 = X_1$, enquanto que $Y_i = D_{i-1} + X_i$ para $i = 2,3,\ldots$. Nessa situação a função de distribuição $H_1(y)$ é

$$H_1(y) = P[Y_1 \leq y] = P[X_1 \leq y] = F_X(y) \tag{2.54}$$

e a função de distribuição $H(y)$ para os demais tempos entre falhas é dada por (2.53).

O número médio de falhas e reparos pode ser obtido pelo uso de transformadas de Laplace. Para maiores detalhes, consultar (RAUSAND; HOYLAND, 2003).

2.4.3 Superposição de Processos de Renovação

Considere um sistema formado por componentes que tenham processos de renovação não necessariamente com a mesma distribuição subjacente. O processo formado pela união de todas as falhas é chamado de *processo de renovação superposto*. A figura 2.11 mostra os processos de renovação individuais de cada componente e a superposição dos processos.

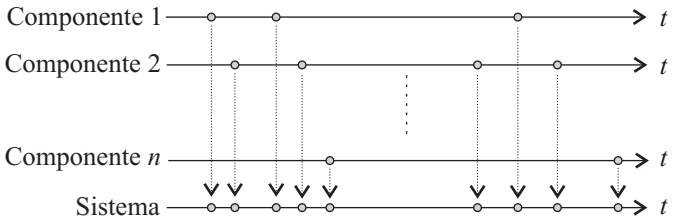

Figura 2.11: Superposição de processos de contagem

Para um sistema formado apenas de componentes em série, se eles forem modelados por HPPs, a superposição desses HPPs será um HPP. Mas, em geral, um processo de renovação superposto não é um processo de renovação. Dessa maneira as distribuições das variáveis aleatórias relacionadas ao sistema (T_r, X_i, $N(t)$), bem como métricas de desempenho (disponibilidade, por exemplo) não podem ser obtidas analiticamente. Assim, recorre-se à simulação.

2.5 Simulação Discreta de Eventos

A simulação discreta de eventos é uma técnica computacional utilizada para compreender o comportamento de sistemas. Ela consiste em gerar eventos aleatórios e discretos ao longo do tempo de missão a fim de criar um cenário "típico" para o sistema e assim poder avaliar algumas de suas características (BANKS *et al.*, 2001).

A geração de números aleatórios de acordo com distribuições específicas é muito importante para a simulação discreta de eventos. Um dos métodos para se obter um número aleatório g segundo a função de distribuição acumulada $F_G(g)$ consiste em gerar um número aleatório u (uniformemente distribuído em $(0,1)$) e calcular $g = F_G^{-1}(u)$, onde F_G^{-1} indica a função inversa de F_G. Esse método é chamado de *método da transformada inversa* e só pode ser aplicado se $F_G(g)$ for uma função monotônica crescente para todo g. Além disso, ele não pode ser usado quando as funções de distribuição acumuladas não possuem forma analítica, como é o caso da distribuição normal. A figura 2.12 ilustra a geração de um número aleatório segundo a distribuição acumulada $F_G(g)$. Para mais detalhes na geração de números aleatórios, recomenda-se (FISHMAN, 2000).

Segundo Ross (2002), em geral há três tipos de variáveis utilizadas na simulação discreta de

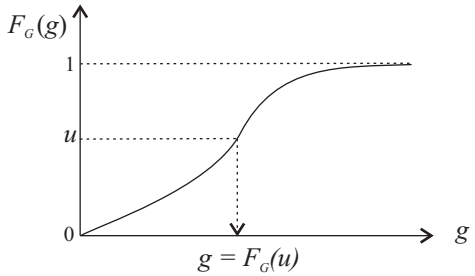

Figura 2.12: Geração inversa de um número aleatório segundo a distribuição $F_G(g)$. Fonte: (RAUSAND; HOYLAND, 2003, p. 377, adaptação)

eventos: *variáveis de tempo*, relativas ao tempo de simulação e aos tempos de ocorrência dos eventos; *variáveis de contagem*, referentes ao número de ocorrências dos eventos de interesse; *variável de estado do sistema*, que indica o estado do sistema em um dado instante de tempo t.

Para um sistema reparável as variáveis de tempo podem estar associadas, por exemplo, aos tempos de falha e aos tempos de reparo. Já a quantidade de falhas e de reparos dos vários componentes pode ser avaliada por meio das variáveis de contagem. Em um determinado instante pode-se obter os estados dos componentes e também do sistema, já que a sua lógica pode ser fornecida pelo seu diagrama de blocos no início da simulação.

Ao final de uma simulação, calculam-se algumas métricas de desempenho do sistema, como a disponibilidade do mesmo no período de tempo simulado (o período em que o sistema esteve operacional dividido pelo tempo total de simulação), o número de falhas e reparos de cada componente, o número de falhas do sistema, entre outras.

O algoritmo a seguir realiza a simulação discreta de eventos para o tempo de missão t_{mis} de um único componente reparável que segue um processo de renovação alternado com tempos entre falhas $X_1, X_2, \ldots$ com funções de distribuição acumuladas $F_{X_i}(x), i = 1, 2, \ldots$ e tempos de reparo $D_1, D_2, \ldots$ com funções de distribuição acumulada $F_{D_i}(d), i = 1, 2, \ldots$. Ele retorna o número de falhas do componente no tempo de missão:

$t \leftarrow 0$, $i \leftarrow 0$
while $t < t_{\text{mis}}$ **do**
  $i \leftarrow i + 1$
  $x_i \leftarrow \text{RAND}(F_{X_i})$ ▷ x_i (tempo de operação até a i-ésima falha) recebe um número aleatório que obedece a distribuição F_{X_i}
  $t \leftarrow t + x_i$
  $d_i \leftarrow \text{RAND}(F_{D_i})$ ▷ d_i (tempo do i-ésimo reparo) recebe um número aleatório que obedece a distribuição F_{D_i}
  $t \leftarrow t + d_i$
return i ▷ número de falhas em t_{mis}

---

3 MÉTODOS DE OTIMIZAÇÃO MULTIOBJETIVO E ALGORITMOS EVOLUCIONÁRIOS

A otimização de um único objetivo pode ser feita por métodos de Programação Matemática bastante consolidados, que dependem da natureza da função-objetivo, das restrições e das variáveis de decisão: programação linear, programação não-linear, programação inteira, entre outros (LUENBERGER, 1984; WOLSEY, 1998). Quando as funções apresentam características como não diferenciabilidade, multimodalidade e espaço de busca complexos, pode-se optar por métodos estocásticos como *simulated annealing* (KIRKPATRICK *et al.*, 1983), algoritmos genéticos, entre outros.

Entretanto, grande parte dos problemas de otimização reais levam em consideração mais de um objetivo, comumente de natureza conflitante. Quer-se, por exemplo, maximizar a confiabilidade ou a disponibilidade ao mesmo tempo em que se deseja minimizar o custo e as perdas de um dado sistema. Dessa maneira, a abordagem multiobjetivo para tratar tais problemas se faz necessária.

De acordo com Yoon e Hwang (1995), os métodos que auxiliam a tomada de decisão envolvendo múltiplos critérios (*Multiple Criteria Decision-Making*) são subdivididos em métodos multiobjetivo (*Multiple Objective Decision-Making*) e métodos multiatributo (*Multiple Attribute Decision-Making*), como mostra a Figura 3.1.

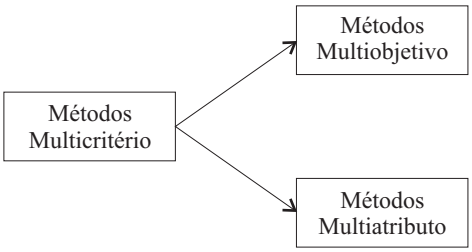

Figura 3.1: Métodos multicritério e suas subdivisões

Todos os métodos multicritério possuem as seguintes características:

- Um conjunto de critérios a serem julgados.
- Um conjunto de variáveis de decisão.
- Um processo de comparação das alternativas.

Existem, no entanto, diferenças entre as abordagens multiobjetivo e multiatributo. Os métodos de otimização multiobjetivo não possuem alternativas pré-determinadas, têm um conjunto de objetivos quantificáveis, um conjunto de restrições e um processo para obter informações

sobre o compromisso entre os objetivos. Já os métodos multiatributo normalmente têm um número limitado de alternativas pré-determinadas, as variáveis de decisão não são necessariamente quantificáveis e a seleção da alternativa se faz por meio de comparações inter e intra-atributos, levando-se em conta a estrutura de preferências do decisor. Para maiores detalhes em métodos multiatributo consultar (VINCKE, 1992), (YOON; HWANG, 1995), (ROY, 1996), (GOMES *et al.*, 2002).

É importante ressaltar que essa terminologia para os termos multicritério, multiobjetivo e multiatributo não é rigorosa e em muitos trabalhos eles são usados como sinônimos.

Levando-se em consideração as diferenças apresentadas entre os métodos multicritério, a confiabilidade / disponibilidade e o custo em projetos de sistema são claramente quantificáveis e a definição de um número pré-determinado de alternativas requer uma quantidade limitada de combinações entre os objetivos. Essa limitação acarreta em perda de possíveis combinações dos objetivos e, para evitar que isso ocorra, opta-se pelo uso de métodos multiobjetivo. Diversos trabalhos de otimização multicritério de problemas relacionados à confiabilidade utilizam a abordagem multiobjetivo, como por exemplo os apresentados em (DHINGRA, 1992), (BUSACCA *et al.*, 2001), (MARSEGUERRA *et al.*, 2005), (LAPA *et al.*, 2006) e (TABOADA; COIT, 2007).

A forma geral de um problema multiobjetivo é apresentada a seguir.

$$\underset{\mathbf{x}}{\text{Maximize}} \quad \mathbf{z} = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \tag{3.1}$$

$$\begin{aligned} \text{Sujeito a} \quad & g_i(\mathbf{x}) = 0, \quad i = 1, \ldots, p \\ & h_i(\mathbf{x}) \leq 0, \quad i = p+1, \ldots, m, \end{aligned}$$

onde $\mathbf{z}$ é o vetor formado por k funções-objetivo, $\mathbf{x}$ é o vetor n-dimensional de variáveis de decisão, p é o número de restrições de igualdade e $m - p$ é o número de restrições de desigualdade.

Em um problema com múltiplos objetivos, pode-se querer maximizar todas as funções-objetivo, minimizar todas as funções-objetivo ou minimizar algumas e maximizar outras. Nesse último caso, a formulação matemática do problema pode ser obtida por meio de uma modificação em (3.1):

$$\underset{\mathbf{x}}{\text{Maximize}} \quad \mathbf{z} = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_l(\mathbf{x}), -f_{l+1}(\mathbf{x}), -f_{l+2}(\mathbf{x}), \ldots, -f_k(\mathbf{x})] \tag{3.2}$$

$$\begin{aligned} \text{Sujeito a} \quad & g_i(\mathbf{x}) = 0, \quad i = 1, \ldots, p \\ & h_i(\mathbf{x}) \leq 0, \quad i = p+1, \ldots, m, \end{aligned}$$

onde $f_h(\mathbf{x})$, $h = 1, 2, \ldots, l$ devem ser maximizadas e $f_h(\mathbf{x})$, $h = l+1, l+2, \ldots, k$ devem ser minimizadas.

3.1 Dominância e Não-Dominância

Antes de prosseguir com a descrição dos métodos de otimização multiobjetivo, é preciso definir os conceitos de dominância e não-dominância (COELLO *et al.*, 2002; KUO *et al.*, 2001; ZITZLER, 1999). Ao contrário da otimização de um único objetivo, que possui apenas uma solução, na abordagem multiobjetivo busca-se um conjunto ótimo de soluções *não-dominadas*. Esse conjunto é também conhecido como conjunto ótimo de Pareto, em homenagem ao economista italiano Vilfredo Pareto que generalizou a idéia de "ótimo" no contexto multiobjetivo (COELLO *et al.*, 2002).

Diz-se que uma solução é não-dominada se para todos os objetivos ela tiver um desempenho pelo menos tão bom quanto o das demais e, pelo menos para um deles, seu desempenho é superior ao das outras soluções. Isto é, para dois vetores de variáveis de decisão $\mathbf{x_1}$ e $\mathbf{x_2}$,

$$\mathbf{x_1} \succ \mathbf{x_2} \iff f_h(\mathbf{x_1}) \geq f_h(\mathbf{x_2}), \forall h \quad \text{e} \quad f_h(\mathbf{x_1}) > f_h(\mathbf{x_2}), \text{ para algum } h, \tag{3.3}$$

onde o sinal $\succ$ indica dominância, $\mathbf{x_1}$ é uma solução não-dominada para um problema de maximização e $\mathbf{x_2}$ é uma solução dominada para o mesmo problema. Caso fosse um problema de minimização, os sinais $\geq$ e $>$ das inequações em (3.3) seriam substituídos por $\leq$ e $<$, nessa ordem.

Os conceitos de ótimo local e global da otimização mono-objetivo, dão lugar, respectivamente, ao *conjunto ótimo de Pareto local* e ao *conjunto ótimo de Pareto global* no caso multiobjetivo (DEB, 1999; ZITZLER, 1999). Para definir esses conceitos, considere $\mathbf{X}$ como sendo o conjunto de todos os $\mathbf{x}$ que satisfazem as restrições em (3.1), isto é, o conjunto viável:

- Conjunto ótimo de Pareto local ($\underline{P}$): para todo $\mathbf{x} \in \underline{P}$, não existe solução $\mathbf{x}' \in \mathbf{X}$ satisfazendo $||\mathbf{x}' - \mathbf{x}|| < \varepsilon$ que domine qualquer membro do conjunto $\underline{P}$ ($||\cdot||$ é distância entre os dois pontos e ε é um número positivo pequeno). As soluções que pertencem a $\underline{P}$ constituem um conjunto ótimo de Pareto local.

- Conjunto ótimo de Pareto global ($\overline{P}$): para todo $\mathbf{x} \in \overline{P}$, não existe $\mathbf{x}' \in \mathbf{X}$ tal que $\mathbf{x}' \succ \mathbf{x}$.

Deb (1999) enfatiza que um conjunto de soluções não-dominadas é definido no contexto de uma amostra do espaço de busca viável, já um conjunto ótimo de Pareto é um conjunto de soluções não-dominadas cuja amostra é igual a todo o espaço de busca viável.

Aplicando-se as soluções pertencentes ao conjunto $\overline{P}$ nas funções-objetivo, obtém-se a *fronteira global de Pareto*. Portanto, $F\overline{P}$ é fronteira global de Pareto se

$$F\overline{P} = \left\{ \mathbf{f} = \left(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}) \right), \forall \mathbf{x} \in \overline{P} \right\}. \tag{3.4}$$

Analogamente, substituindo-se os vetores de variáveis de decisão constituintes do conjunto

$\underline{P}$ nas funções-objetivo, obtém-se a *fronteira local de Pareto* (Figura 3.2).

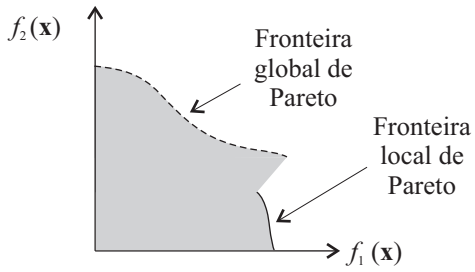

Figura 3.2: Fronteiras local e global de Pareto. Fonte: (ZITZLER, 1999, p. 9, adaptação)

A Figura 3.3 mostra o mapeamento do conjunto $\overline{P}$, contido no espaço das variáveis, no conjunto $F\overline{P}$, contido no espaço das funções-objetivo para um problema com $n = 2$ e $k = 2$.

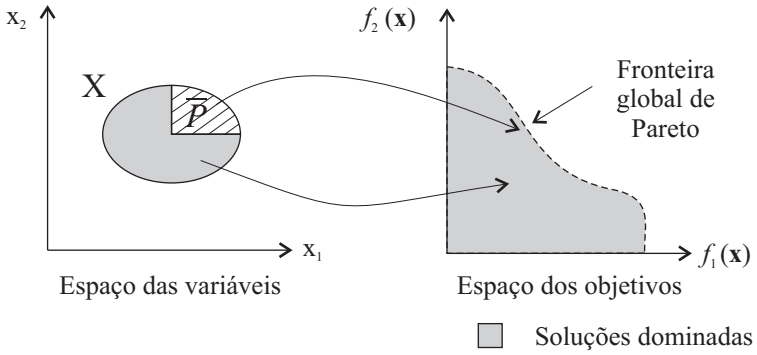

Figura 3.3: Mapeamento do conjunto de soluções não-dominadas na fronteira de Pareto

3.2 Classificação dos Métodos Multiobjetivo

Na abordagem multiobjetivo existem várias soluções potenciais, ao contrário da mono-objetivo que tem apenas uma única – a solução ótima. A otimização multiobjetivo possui duas etapas distintas: a busca de soluções potenciais e a decisão por uma das soluções que apresente o compromisso *adequado* entre os objetivos. Assim, a escolha de *uma* solução está associada às preferências do decisor. A maneira como essas duas etapas se combinam classificam os métodos multiobjetivo em três categorias (ZITZLER, 1999):

- Decisão antes da busca: os objetivos são agregados em uma única função-objetivo, que inclui implicitamente informações sobre preferências fornecidas pelo decisor.
- Decisão durante a busca: o decisor fornece informações sobre suas preferências durante o processo de busca; a cada iteração elas são usadas para determinar novas soluções. As

preferências do decisor guiam o processo de busca. Os métodos dessa categoria também são conhecidos como métodos interativos.

- Busca antes da decisão: o processo de otimização é feito sem qualquer informação sobre as preferências do decisor, fornece como resultado um conjunto de possíveis soluções e só então o decisor faz sua escolha.

A primeira categoria de métodos transforma o problema multiobjetivo em mono-objetivo a partir das preferências do decisor, possibilitando o uso de técnicas de otimização de um objetivo único. Apesar de não analisarem explicitamente as preferências do decisor, Lapa *et al.* (2006) utilizam a agregação das funções-objetivo por meio de pesos para encontrar políticas de manutenção preventiva baseadas em custo e confiabilidade. Nos casos em que a estrutura das preferências do decisor são consideradas de maneira explícita, a forma como elas são incorporadas ao problema é bastante relevante.

O segundo conjunto de métodos exige participação do decisor no processo de otimização, o que nem sempre é possível. Ainda que o decisor esteja disponível, é preciso que ele tenha conhecimento aprofundado sobre o problema em questão e também que ele esteja familiarizado com o processo de otimização, pois é ele quem fornece as diretrizes de tal processo. Caso contrário, o resultado encontrado pode não ser satisfatório. Nesses métodos, as características e restrições físicas do problema, bem como as preferências do decisor são consideradas simultaneamente. Um método inserido nessa categoria é, por exemplo, desenvolvido em (TAPPETA; RENAUD, 2001) e aplicado em um problema de projeto mecânico.

Os métodos da terceira categoria consideram apenas as restrições físicas (KUO *et al.*, 2001) do problema e não requerem informações prévias sobre as preferências do decisor. Eles buscam encontrar um conjunto de soluções equivalentes do ponto de vista multiobjetivo e o decisor pode então optar por uma delas, com base em suas preferências. No entanto, em alguns casos, o conjunto de soluções encontrado pode ser muito grande e/ou de difícil visualização (nos casos de mais de três objetivos, por exemplo), dificultando a escolha de uma única alternativa por parte do decisor. Das três categorias, essa é a única que requer participação do decisor *após* a busca por soluções potenciais. Isso, de certa forma, evita problemas associados à falta de preparo dos decisores em relação aos processos de elicitação de preferências e de otimização freqüentemente requeridos por métodos das demais categorias. A literatura apresenta alguns trabalhos que usam essa abordagem em problemas multiobjetivo relacionados à confiabilidade: Dhingra (1992), por exemplo, usa programação por metas (ARENALES *et al.*, 2007) para encontrar soluções ótimas de Pareto em um problema de alocação de confiabilidade e redundâncias em um sistema em série; Busacca *et al.* (2001), Marseguerra *et al.* (2005) e Taboada e Coit (2006) utilizam algoritmos genéticos multiobjetivo para encontar soluções ótimas de Pareto em problemas de alocação de redundâncias.

Na prática, o conhecimento das próprias preferências e do problema em questão por parte do decisor nem sempre é suficiente para que ele indique a importância relativa entre os objetivos ou possa participar do processo de otimização . Além disso, no problema de alocação de redundâncias, o número de possíveis soluções, em geral, é grande e assim pode haver soluções não imaginadas pelo decisor ainda que este conheça o problema e esteja consciente de suas próprias preferências. Dessa maneira, optou-se por elaborar um método multiobjetivo que se insere na categoria de métodos que realizam a busca antes da decisão.

Coello *et al.* (2002) fazem um resumo dos principais métodos de cada uma das três categorias apresentadas. A próxima seção introduz os dois principais métodos multiobjetivo tradicionais representantes da terceira categoria.

3.3 Métodos Tradicionais

Os principais métodos de otimização multiobjetivo nos quais a busca de soluções é feita antes da decisão são o Método da Soma Ponderada e o Método da ε-Perturbação (COELLO *et al.*, 2002).

3.3.1 Método da Soma Ponderada

Nesse método, as funções-objetivo são combinadas em uma única função-objetivo, como apresentado na equação (3.5).

$$\underset{\mathbf{x}}{\text{Maximize}} \quad \mathbf{z} = \sum_{j=1}^{k} \omega_j f_j(\mathbf{x}) \tag{3.5}$$

$$\begin{aligned} \text{Sujeito a} \qquad g_i(\mathbf{x}) &= 0, \quad i = 1, \ldots, p \\ h_i(\mathbf{x}) &\leq 0, \quad i = p+1, \ldots, m \\ \omega_j &\geq 0, \quad \forall j \\ \omega_j &> 0, \quad \text{para algum } j, \\ \textstyle\sum_{j=1}^{k} \omega_j &= 1, \end{aligned}$$

em que ω_j é o peso do j-ésimo objetivo.

Essa agregação dos objetivos também caracteriza métodos em que a decisão é realizada antes da busca por soluções. No entanto, a aplicação desse método usando uma abordagem ou outra apresenta algumas diferenças. Em geral, a solução de (3.5) é não-dominada e pertence ao conjunto ótimo de Pareto. Se a decisão for posterior à busca, para se obter diferentes soluções ótimas de Pareto, variam-se os pesos ω_j na função-objetivo e resolve-se o problema mono-objetivo resultante em cada caso. Além disso, os pesos ω_j não representam a importância

relativa entre os objetivos, são apenas fatores que ao variarem localizam pontos no conjunto ótimo de Pareto (COELLO *et al.*, 2002). Por outro lado, se a soma ponderada for usada como um método de decisão anterior à busca, os pesos ω_j normalmente representam a importância relativa entre os objetivos para o decisor e o problema é resolvido uma única vez, obtendo-se uma única solução.

Fixando-se o vetor de pesos, a otimização da equação (3.5) consiste, graficamente, em encontrar um hiperplano com orientação fixa no espaço dos objetivos e "deslizá-lo" até que se encontre um ponto em que ele – o hiperplano – e o espaço de busca viável tenham tangente comum, isto é, não pode haver soluções viáveis acima (problema de maximização) ou abaixo (problema de minimização) e pelo menos uma solução viável deve estar no hiperplano (ZITZLER, 1999). No caso de um problema com dois objetivos, o hiperplano é uma reta.

Nota-se que a função-objetivo em (3.5) é uma combinação linear (BOLDRINI *et al.*, 1980) das múltiplas funções-objetivo em que a soma dos coeficientes é igual à unidade, logo a função-objetivo é uma *combinação convexa* das diversas funções-objetivo.

Uma das principais desvantagens desse método é o fato de ele não conseguir encontrar soluções não-dominadas em determinadas regiões da fronteira ótima de Pareto se ela for não-convexa (MESSAC *et al.*, 2000). Para melhor entender esse aspecto do método da soma ponderada, a seguir são mostradas as definições gráficas de *conjunto convexo* e *função convexa* (HILLIER; LIEBERMAN, 1967):

- Conjunto convexo: um conjunto é convexo se ele contém completamente o segmento de reta que une quaisquer de seus dois pontos. Caso contrário, trata-se de um *conjunto não-convexo* ou *côncavo* (Figura 3.4).

- Função convexa: uma função é convexa se o segmento de reta que une quaisquer dois pontos de seu gráfico está completamente acima dele ou sobre ele. Se o segmento de reta estiver completamente abaixo do gráfico ou sobre ele, diz-se que a função é *não-convexa* ou *côncava* (Figura 3.5).

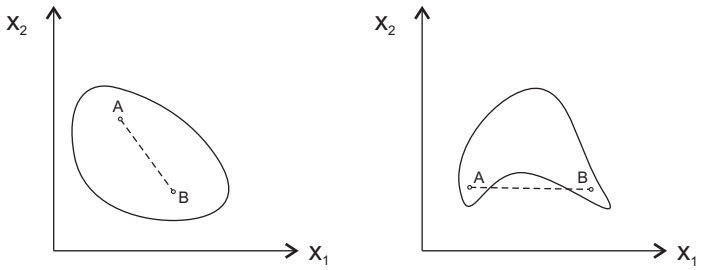

Figura 3.4: Conjunto convexo e conjunto não-convexo

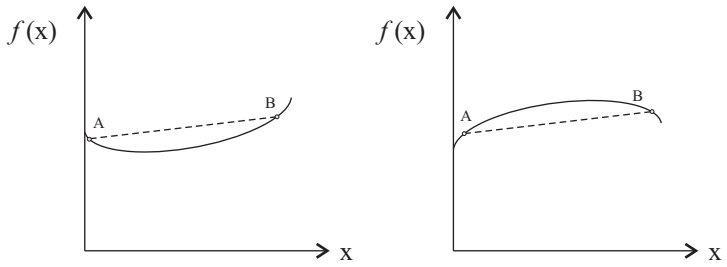

Figura 3.5: Função convexa e função não-convexa

A Figura 3.6 mostra uma fronteira ótima de Pareto convexa e a reta encontrada para dois pesos fixos ω_1 e ω_2 para o caso de dois objetivos a serem maximizados. Observa-se que ao se variar os pesos (e conseqüentemente a inclinação da reta), pode-se obter os pontos B e C, por exemplo. Já a Figura 3.7 ilustra uma fronteira ótima de Pareto não-convexa. Para se obter o ponto A (contido na região côncava), a reta teria de interceptar a fronteira em dois pontos – o que não caracterizaria uma tangente à região viável – e as soluções B e C estariam acima dela, o que não pode ocorrer. Para qualquer combinação de pesos apenas as soluções B e C seriam encontradas e as soluções ótimas de Pareto contidas na região côncava não seriam obtidas. Para maiores detalhes sobre o assunto consultar (DAS; DENNIS, 1997) e (MESSAC *et al.*, 2000).

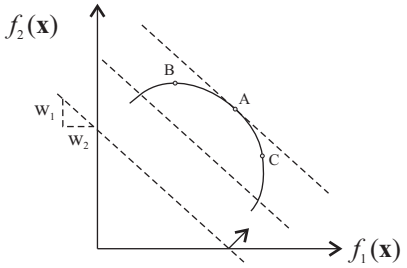

Figura 3.6: Fronteira ótima de Pareto convexa. Fonte: (DEB, 1999, p. 6, adaptação)

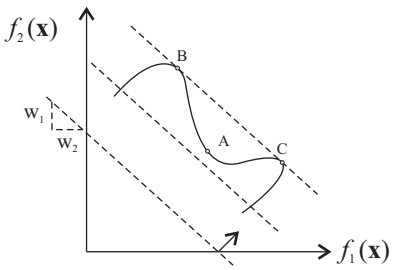

Figura 3.7: Fronteira ótima de Pareto não-convexa. Fonte: (DEB, 1999, p. 6, adaptação)

3.3.2 Método da ε-Perturbação

Nesse método, um objetivo escolhido arbitrariamente (ou o considerado mais relevante) é otimizado e os demais se tornam restrições do problema (3.6).

$$\underset{\mathbf{x}}{\text{Maximize}} \quad \mathbf{z} = f_l(\mathbf{x}) \tag{3.6}$$

$$\begin{aligned} \text{Sujeito a} \quad & g_i(\mathbf{x}) = 0, \quad i = 1,\ldots,p \\ & h_i(\mathbf{x}) \leq 0, \quad i = p+1,\ldots,m, \\ & f_j(\mathbf{x}) \geq \varepsilon_j, \quad j = 1,2\ldots,k, \text{ e } j \neq l, \end{aligned} \tag{3.7}$$

em que ε_j é um nível aceitável previamente determinado para o j-ésimo objetivo.

Esse método, ao contrário do método da soma ponderada, é capaz de encontrar soluções nas regiões côncavas de fronteiras ótimas de Pareto não-convexas, como é ilustrado na Figura 3.8.

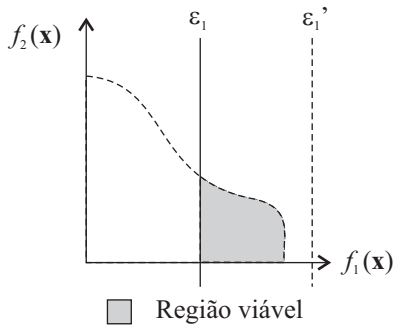

Figura 3.8: Método da ε-perturbação para dois objetivos a serem maximizados

Variando-se os valores de ε_j, obtêm-se múltiplas soluções ótimas de Pareto. É importante notar que o estabelecimento de níveis mínimos aceitáveis ε_j para as restrições (3.7) requer uma análise preliminar, já que a escolha inapropriada desses níveis pode resultar em conjuntos viáveis vazios (na Figura 3.8, se $f_1(\mathbf{x}) \geq \varepsilon_1{}'$, não há região viável). Uma maneira de obter valores iniciais de ε_j é resolver um problema de otimização mono-objetivo para cada uma das funções-objetivo.

Uma das maiores desvantagens dos métodos tradicionais consiste no fato de que eles retornam uma única solução por vez. Isso implica que, para se obter diferentes soluções ótimas de Pareto (ou não), o algoritmo deve ser executado diversas vezes. Além disso, a eficácia do método da soma ponderada depende da convexidade da fronteira de Pareto e o método da ε-perturbação requer um conhecimento prévio sobre intervalos viáveis para os níveis aceitáveis de cada objetivo. Isto é, antes de serem aplicados, deve-se ter um considerável conhecimento sobre o problema. Uma outra questão comentada em (DEB, 1999), é a de os médotos clássicos não serem eficientes para resolver problemas que possuem espaços de busca discretos.

Alternativamente aos métodos clássicos, algoritmos evolucionários podem ser usados para resolver problemas multiobjetivo. Na verdade, eles superam a maior parte das dificuldades dos métodos tradicionais. Algoritmos evolucionários lidam com várias soluções potenciais simultaneamente, o que permite encontrar diferentes soluções ótimas de Pareto em uma única execução do algoritmo. Além disso, não apresentam problemas com relação à convexidade e à continuidade da fronteira de Pareto.

A próxima seção descreve os fundamentos de algoritmos evolucionários, mais especificamente algoritmos genéticos, cita suas principais abordagens multiobjetivo e descreve brevemente duas metodologias usadas no problema de alocação de redundâncias.

3.4 Algoritmos Evolucionários

Os algoritmos evolucionários fazem parte de uma classe de métodos de otimização estocásticos denominada computação evolucionária, que simulam computacionalmente o processo de evolução natural. Os primeiros trabalhos relacionados à computação evolucionária surgiram no final da década de 1950, no entanto o seu desenvolvimento efetivo iniciou-se nos anos de 1970.

Os algoritmos evolucionários (assim como os demais métodos compreendidos pela computação evolucionária) são direcionados, principalmente, à otimização de problemas que apresentam características não toleráveis pelos métodos clássicos de programação matemática. Além disso, tais algoritmos possuem características bastante favoráveis para serem utilizados na resolução de problemas multiobjetivo.

3.4.1 Fundamentos de Algoritmos Evolucionários

Por serem baseados no processo de evolução natural, a nomenclatura utilizada pelos algoritmos evolucionários origina-se da biologia. Na abordagem de algoritmos evolucionários, para resolver um problema mono-objetivo, necessita-se inicialmente de uma *representação* para os vetores de variáveis de decisão (as soluções potenciais). Essa representação geralmente é binária ou real. Cada variável codificada representa um *gene* e o conjunto de genes forma o *genótipo*. Quando o genótipo é decodificado para a representação original, tem-se o *fenótipo*. No caso da representação real, genótipo e fenótipo são iguais. Um *indivíduo* é uma solução potencial do problema e possui um genótipo e um fenótipo a ele associado. A Figura 3.9 mostra o processo de codificação e decodificação de um indivíduo para a representação binária. O conjunto de indivíduos forma uma *população*.

Em seguida, operadores genéticos como seleção, *crossover* (ou *recombinação*) e *mutação* são aplicados na população.

Os indivíduos são selecionados com base nos ajustes que apresentam. O *ajuste* ou *fitness*

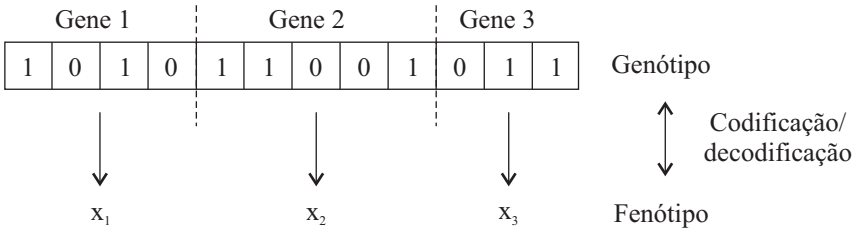

Figura 3.9: Representação de um indivíduo

de cada indivíduo é medido substituindo-se o seu respectivo fenótipo na *função de ajuste*, que no caso de otimização de problemas numéricos geralmente é igual à função-objetivo (COELLO, 1999). Os indivíduos que apresentam melhores ajustes têm mais chances de serem selecionados, enquanto os mais "fracos", que apresentam piores ajustes, são descartados. O intuito da seleção é restringir a busca a determinadas regiões do espaço de busca e melhorar a qualidade média da população (ZITZLER, 1999).

Quando o problema apresenta restrições, se o indivíduo não as satisfizer, incorpora-se uma *penalização* a seu ajuste, por meio da *função de penalidade*. Se o problema é de maximização, o ajuste é diminuído de um certo valor. Analogamente, se o problema é de minimização, o ajuste é piorado por meio da adição de um certo valor. Nesse caso, os indivíduos são selecionados de acordo com o *ajuste penalizado* que apresentam.

Quanto maior a pressão da seleção, menor a diversidade de indivíduos, e conseqüentemente a exploração do espaço de busca como um todo pode ser comprometida e acarretar uma convergência prematura do algoritmo para um ótimo local. No entanto, é preciso haver uma certa pressão de seleção para que a convergência não seja lenta demais.

Os indivíduos selecionados para *crossover* (os *pais*), de acordo com a probabilidade de *crossover* estabelecida, têm seus genótipos combinados para gerarem *filhos*. Em seguida, a população e os indivíduos resultantes do *crossover* são submetidos à mutação, isto é, parte do genótipo de alguns indivíduos é modificada, segundo uma probabilidade de mutação associada. Tanto o *crossover* quanto a mutação buscam gerar novas soluções dentro do espaço de busca por meio da variação das soluções já existentes. É importante notar que os operadores de *crossover* e mutação atuam nos genótipos, enquanto a seleção é baseada na avaliação dos fenótipos.

Na maior parte dos algoritmos, a população tem tamanho fixo, então é necessário descartar alguns indivíduos em favor de outros. Essa etapa é chamada de *substituição*. As técnicas mais comuns são a de *pais darem lugar aos filhos* – os indivíduos participantes do *crossover* são descartados e cedem lugar aos indivíduos por eles gerados, sem considerar os ajustes – e a *substituição elitista* – após a mutação os ajustes da população formada por pais e filhos são avaliados, os indivíduos são ordenados e os N primeiros formam a nova população, onde N é o tamanho fixo da população.

A simulação do processo evolutivo começa com a geração aleatória, ou de acordo com uma estratégia pré-definida, da população inicial. Avalia-se o ajuste (penalizado, se houver restrições) dos indivíduos dessa população, aplica-se a seleção, a recombinação e/ou a mutação e a substituição. Assim obtém-se uma nova população e uma iteração do algoritmo é finalizada. Repete-se a avaliação do ajuste dos indivíduos, a aplicação dos operadores genéticos e a substituição até que um critério de parada seja alcançado. Cada repetição dessa é chamada de *geração* e normalmente o número máximo de gerações constitui o critério de parada do algoritmo. Outros critérios de parada podem ser usados, como, por exemplo, a similaridade dos indivíduos da população. O indivíduo ótimo deve ser atualizado a cada geração e o resultado desse processo é o melhor indivíduo gerado ao longo de toda a evolução. A Figura 3.10 ilustra o fluxograma para algoritmos genéticos com tamanho de população fixo.

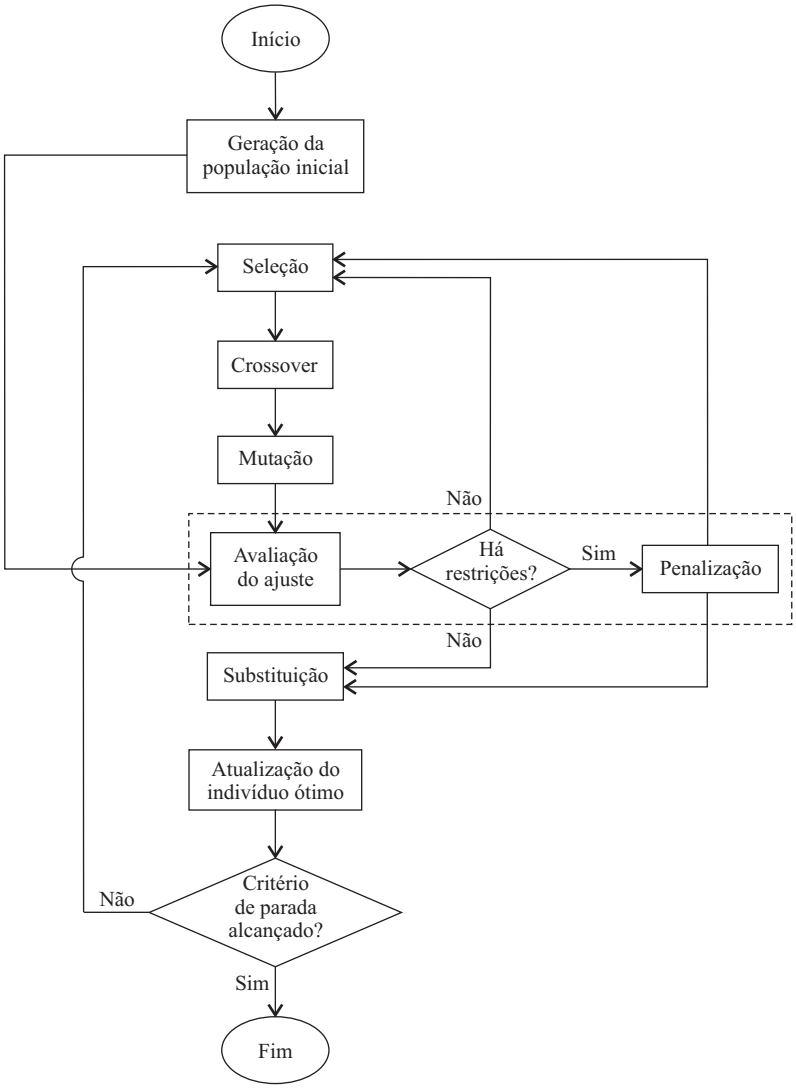

Figura 3.10: Fluxograma de Algoritmos Genéticos para um objetivo único

A principal desvantagem da otimização via algoritmos evolucionários é a a falta de garantia de que eles dêem como saída a solução ótima. No entanto, pode-se validar o algoritmo através da aplicação de problemas existentes na literatura e comparação de resultados. Além disso, os problemas complexos que os algoritmos evolucionários podem "tentar" resolver muitas vezes não podem ser solucionados por meio de métodos clássicos de programação matemática. Assim, mesmo que não se obtenha uma solução ótima, *boas* soluções podem ser encontradas e aplicadas.

De acordo com Bäck *et al.* (1997), os algoritmos evolucionários (AE) compreendem os algoritmos genéticos (AG), a programação evolucionária (PE) e as estratégias evolucionárias (EE). Essas três abordagens, apesar de apresentarem características comuns, foram desenvolvidas independentemente. As principais diferenças entre elas são mostradas na Tabela 3.1, onde μ representa o número de pais e λ, o número de filhos. Em linhas gerais, na seleção-$(\mu+\lambda)$ os μ melhores indivíduos da população formada por pais e filhos são selecionados. Já na seleção-(μ,λ), os μ melhores indivíduos apenas da população de filhos são escolhidos.

Tabela 3.1: Principais diferenças entre os algoritmos evolucionários. Fonte: (COELLO *et al.*, 2002, p. 24, adaptação)

AE	**Representação das variáveis**	**Operadores Evolucionários**
AG	Binária, inteira ou real	Seleção, recombinação e mutação
PE	Real	Seleção-$(\mu+\lambda)$ ou (μ,λ), recombinação e mutação
EE	Real e parâmetros de estratégia	Seleção-$(\mu+\lambda)$ e mutação

Nesse trabalho, dá-se ênfase aos algoritmos genéticos por serem bastante flexíveis para trabalharem com a abordagem multiobjetivo, além de originalmente terem sido desenvolvidos para a otimização. Para maiores detalhes em programação evolucionária e em estratégias de evolução consultar (FOGEL *et al.*, 1995) e (BEYER; SCHWEFEL, 2002), respectivamente.

3.4.2 Algoritmos Evolucionários Multiobjetivo

Em vez de considerar apenas uma função-objetivo, na abordagem multiobjetivo quer-se resolver duas ou mais funções-objetivo simultaneamente. Como foi mencionado anteriormente, para encontrar soluções ótimas de Pareto, os métodos tradicionais precisam ser executados diversas vezes. Assim, os algoritmos evolucionários apresentam uma vantagem fundamental em relação aos métodos tradicionais, já que são baseados em uma população de soluções potenciais e por isso pode-se encontrar várias soluções não-dominadas que pertencem ao conjunto ótimo de Pareto numa única iteração. As principais metodologias desenvolvidas para a otimização multiobjetivo via algoritmos evolucionários são:

- *Vector Evaluated Genetic Algorithms* (VEGA), por Schaffer (1985).

- *Multi-Objective Genetic Algorithm* (MOGA), por Fonseca e Fleming (1993).
- *Niched-Pareto Genetic Algorithm* (NPGA), por Horn *et al.* (1994).
- *Nondominated Sorting Genetic Algorithm* (NSGA), por Srinivas e Deb (1994).
- *Strength Pareto Evolutionary Algorithm* (SPEA), por Zitzler e Thiele (1999).
- *Nondominated Sorting Genetic Algorithm II* (NSGA-II), por Deb *et al.* (2002).

A principal diferença de um algoritmo evolucionário mono-objetivo para um multiobjetivo é a etapa de seleção. Como há duas ou mais funções-objetivo, o conceito de não-dominância é usado direta ou indiretamente (por exemplo, nos algoritmos NSGA e VEGA, respectivamente). Além disso, opta-se freqüentemente por um ajuste elaborado dos indivíduos, que seja capaz de incorporar as características por eles apresentadas nos diversos objetivos. Esses ajustes elaborados podem ser, por exemplo, baseados no número de soluções que dominam cada indivíduo, de forma que os indivíduos não-dominados possuam melhor valor de ajuste e que os indivíduos dominados pelo maior número de soluções receba o pior ajuste, como é o caso do MOGA. Já no NSGA-II, os indivíduos são avaliados de acordo com a dominância e não-dominância, mas também segundo as distâncias que apresentam uns em relação aos outros no espaço dos objetivos. Nesse caso um indivíduo é preferível a outro se for não-dominado e se ambos forem não-dominados, prefere-se aquele que possui a maior distância em relação às demais soluções. É interessante que as soluções que apresentam maiores distâncias sejam preferíveis para que a fronteira de Pareto possa ser explorada ao máximo.

Marseguerra *et al.* (2005) aplicam algoritmos genéticos multiobjetivo juntamente com simulação Monte Carlo para alocação de redundâncias. Eles utilizam uma estratégia de ordenação da população com base no número de soluções que dominam cada indivíduo, de forma semelhante ao MOGA. Durante a busca, um arquivo (de tamanho fixo pré-definido) contendo soluções não-dominadas que representam dinamicamente a fronteira ótima de Pareto é atualizado. Ao fim de cada geração, as soluções não-dominadas da população corrente são comparadas com as já existentes no arquivo e as seguintes regras devem ser satisfeitas:

- Se uma nova solução domina indivíduos do arquivo, esses são eliminados e se acrescenta a nova solução.
- Se uma nova solução é dominada por qualquer indivíduo do arquivo ela não é guardada.
- Se uma nova solução nem domina nem é dominada por indivíduos do arquivo então:
  - Se o arquivo não estiver cheio, ela é guardada.

– Se o arquivo estiver cheio, ela substitui a solução do arquivo mais semelhante (essa semelhança é avaliada a partir das distâncias euclidianas entre os valores dos ajustes apresentados pelos indivíduos).

Taboada e Coit (2006) desenvolveram um algoritmo evolucionário, com base no NSGA-II, especialmente para resolver o problema de alocação de redundâncias, o MOEA-DAP (*Multiple Objective Evolutionary Algorithm for Solving Design Allocation Problems*). O método utiliza duas métricas de ajuste, uma para promover a diversidade e outra para contribuir com a aproximação dos indivíduos: a primeira é baseada na distância euclidiana no espaço dos objetivos que um determinado indivíduo apresenta em relação aos outros indivíduos e a segunda considera o número de soluções que ele domina. Essas duas métricas são somadas para formar o *ajuste agregado* do indivíduo e quanto maior o valor desse ajuste, melhor será o indivíduo.

A cada geração, os indivíduos dominados são eliminados, permitindo que a população seja formada apenas por indivíduos não-dominados. Vale ressaltar que nos exemplos mostrados em (TABOADA; COIT, 2006), apesar de existirem restrições, indivíduos inviáveis não são gerados e conseqüentemente não há a necessidade de penalizar os indivíduos. Além disso, na formação da nova população, usa-se uma estratégia elitista – uma porcentagem dos melhores indivíduos seguem para a próxima geração – e portanto é preciso ordenar os indivíduos com relação ao ajuste agregado que eles apresentam. A saída do algoritmo é a população da última geração.

O próximo capítulo descreve a metodologia proposta para a resolução de problemas de alocação de redundâncias baseadas em um algoritmo genético multiobjetivo.

4 ALGORITMO GENÉTICO MULTIOBJETIVO PROPOSTO PARA RESOLVER PROBLEMAS DE ALOCAÇÃO DE REDUNDÂNCIAS EM CONFIABILIDADE

Nesse capítulo é descrito o algoritmo genético multiobjetivo desenvolvido durante a realização desse trabalho em conjunto com o grupo de pesquisa RISCTEC-UFPE. Ele foi elaborado para resolver o problema de alocação de redundâncias em sistemas não-reparáveis e reparáveis, segundo uma abordagem multiobjetivo em que se busca maximizar a confiabilidade ou a disponibilidade e minimizar os custos associados. Considera-se que os sistemas possuem configuração paralelo-série e que existem restrições nas quantidades mínima e máxima de componentes que podem constituir cada um dos subsistemas do sistema em questão.

O problema de alocação de redundâncias considerado consiste em escolher os *tipos* e a *quantidade* de componentes para cada subsistema, de forma que o sistema como um todo tenha simultaneamente uma alta confiabilidade ou disponibilidade e um baixo custo. Assim, as *variáveis de decisão* do problema são *as quantidades de cada tipo de componente* que devem ser alocadas em cada subsistema.

Quando se leva em conta sistemas não-reparáveis, tanto a confiabilidade quanto os custos podem ser calculados de forma analítica. Dessa maneira, os ajustes dos indivíduos relativos a cada um dos objetivos podem ser obtidos diretamente da avaliação dos seus fenótipos nas funções-objetivo.

No caso de sistemas reparáveis, supôs-se que eles são formados por componentes reparáveis cujos processos de falha-reparo são regidos por processos de renovação alternados. O sistema possui, portanto, um processo de falha-reparo que não é caracterizado por um processo de renovação. Assim, nessas situações, o ajuste relativo à disponibilidade é encontrado por meio de simulação discreta de eventos. Além disso, o número médio de reparos por componente utilizado no cálculo dos custos também é obdito por meio de simulação.

A metodologia proposta teve como inspiração tanto o algoritmo usado por Marseguerra *et al.* (2005) quanto o MOEA-DAP (TABOADA; COIT, 2006). Do algoritmo apresentado em (MARSEGUERRA *et al.*, 2005), foi utilizada a estratégia de guardar os indivíduos não-dominados de cada geração em uma população auxiliar. Assim, garante-se que ao final de todas as iterações, os melhores indivíduos gerados ao longo de todo o processo de otimização estarão guardados, independentemente de eles terem sido descartados ou não durante a evolução.

Do MOEA-DAP, utilizou-se o fato de as soluções dominadas serem eliminadas a cada iteração do algoritmo. Ao se adotar essa estratégia, à medida que a evolução prossegue, os indivíduos ficam cada vez mais próximos da fronteira ótima de Pareto, o que é o principal intuito dos métodos multiobjetivo que realizam a busca antes da decisão.

É importante ressaltar que não é utilizado nenhum tipo de ordenação das soluções durante o

processo de otimização e conseqüentemente não é necessária a elaboração de um ajuste único, como é feito, por exemplo, em (DEB *et al.*, 2002), (MARSEGUERRA *et al.*, 2005) e (TABOADA; COIT, 2006). Dessa maneira, cada indivíduo possui um número de ajustes igual ao número de objetivos considerados no problema. Além disso, diferentemente do MOEA-DAP, permite-se que indivíduos inviáveis sejam gerados e estes são penalizados de acordo com uma função de penalização dinâmica, que será descrita mais adiante. Espera-se que indivíduos inviáveis introduzam variabilidade à população, o que possibilita uma maior exploração do espaço de busca e conseqüentemente da fronteira de Pareto.

A seguir são descritas as várias etapas do algoritmo genético multiobjetivo proposto.

4.1 Etapas do Algoritmo Genético Multiobjetivo Proposto

Considere N o tamanho constante da população P, P_{aux} a população auxiliar que guarda os indivíduos não-dominados gerados em cada iteração do algoritmo e $\mathbf{x}$ o vetor de n variáveis de decisão.

4.1.1 Representação dos Indivíduos

O problema de alocação de redundâncias considerado possui variáveis de decisão inteiras e por isso optou-se pela codificação binária para os indivíduos. Se as variáveis de decisão fossem reais, seria mais conveniente usar a codificação real (HERRERA *et al.*, 1998). A representação binária de um número inteiro positivo x pode ser obtida por meio do seguinte algoritmo:

---

procedure BIN(x)
  $i \leftarrow 0$
  while $x \neq 0$ **do**
    $b_i \leftarrow x \mod 2, \quad x \leftarrow \lfloor x/2 \rfloor, \quad i \leftarrow i+1$
  return $b_0, b_1, \ldots, b_{i-1}$

Para obter novamente o valor inteiro x a partir do vetor binário $\langle b_0, b_1, \ldots, b_{i-1} \rangle$ basta aplicar a equação:

$$x = \sum_{j=0}^{i-1} b_j \cdot 2^{i-j+1}. \tag{4.1}$$

Para ilustrar a representação dos indivíduos, suponha que o sistema considerado é formado de dois subsistemas em série, S_1 e S_2. Cada um desses subsistemas pode ter no mínimo 1 e no máximo 4 componentes em paralelo. Além disso, considere que o primeiro subsistema tem três opções de componentes (a_1, a_2 e a_3) e o segundo subsistema possui quatro alternativas de

componentes (e_1, e_2, e_3 e e_4). A figura 4.1 mostra o fenótipo e o genótipo de um indivíduo modelado de acordo com essas suposições. O fenótipo é formado por sete variáveis. As três primeiras representam o número de opções de componentes para S_1 e as quatro últimas representam o número de opções de componentes para S_2. O valor de cada variável indica o número de componentes de cada tipo.

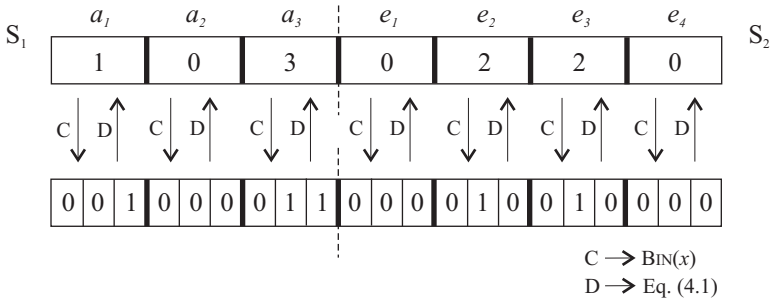

Figura 4.1: Exemplo de fenótipo e genótipo de um indivíduo

O sistema representado pelo indivíduo da figura 4.1 possui um componente a_1 e três componentes a_3 em S_1 e dois componentes e_2 e dois componentes e_3 em S_2. Esse sistema é ilustrado na figura 4.2

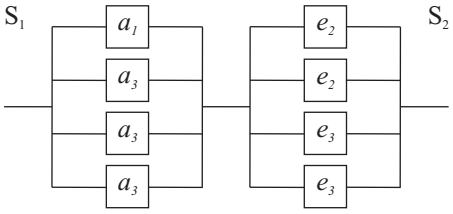

Figura 4.2: Configuração do sistema representado pelo indivíduo da figura 4.1

Essa representação é igual à utilizada por Taboada e Coit (2006).

4.1.2 Geração da População Inicial

Para formar a população inicial, geram-se N indivíduos aleatoriamente. Define-se um limite inferior ℓ e um limite superior u para cada uma das variáveis de decisão. Se um indivíduo $P[i]$ é formado por **x**, a geração aleatória do mesmo é descrita pelo algoritmo a seguir.

---

procedure INDRAND(ℓ_1, u_1, ℓ_2, u_2, ..., ℓ_n, u_n)
  for $j = 1 \ldots n$ **do**
    $x_j \leftarrow$ RAND($\{\ell_j, \ell_j + 1, \ldots, u_j\}$) ▷ escolha aleatória de um inteiro dentro dos limites ℓ_j e u_j
  return $x_1, x_2, \ldots, x_n$ ▷ devolver indivíduo escolhido aleatoriamente

---

4.1.3 Cálculo dos Ajustes e das Violações e Penalização de Indivíduos

Os ajustes dos indivíduos devem ser calculados para cada função-objetivo $f_1, \ldots, f_k$. Em seguida, verifica-se a viabilidade dos indivíduos por meio do cálculo das violações. A notação $v_i(\mathbf{x})$ representa a violação do indivíduo formado por $\mathbf{x}$ com relação à i-ésima restrição, que é dada por:

$$v_i(\mathbf{x}) = \max\{0, |g_i(\mathbf{x})| - \varepsilon\}, \quad \text{se } 1 \leq i \leq p \tag{4.2}$$

$$v_i(\mathbf{x}) = \max\{0, h_i(\mathbf{x})\}, \quad \text{se } p+1 \leq i \leq m, \tag{4.3}$$

em que m é o número total de restrições, p é o número de restrições de igualdade, $m-p$ é o número de restrições de desigualdade e ε é a tolerância que se dá para as restrições de igualdade (precisão da restrição de igualdade). A expressão (4.3) deve ser usada para restrições com sinais $\leq$ ou $<$. Caso a restrição de desigualdade tenha sinal $\geq$ ou $>$, antes de avaliar (4.3), ela deve ser transformada para uma restrição com sinal $\leq$ ou $<$.

As violações são utilizadas no cálculo da penalização. Adota-se uma função de penalização dinâmica (JOINES; HOUCK, 1994):

$$pen(\mathbf{x}) = (c \cdot ger)^{\alpha} \cdot \sum_{i=1}^{m} v_i(\mathbf{x})^{\beta}, \tag{4.4}$$

em que $c = 0,5$, $\alpha = \beta = 2$, m é o número de restrições do problema e *ger* é a geração.

A função de penalização em (4.4) é dita dinâmica porque depende do número de gerações. No início do processo de evolução é mais conveniente permitir maiores violações pois indivíduos inviáveis podem gerar indivíduos viáveis nas próximas gerações e permitem uma maior exploração do espaço de busca (MARTORELL *et al.*, 2000). No entanto, à medida que o algoritmo prossegue, é desejável que a penalização para indivíduos inviáveis seja maior, a fim de que, ao final, tenha-se o maior número possível de indivíduos viáveis.

Os ajustes penalizados de um indivíduo $P[i]$ formado por $\mathbf{x}$ são dados por:

$$f_j^{\text{pen}}(\mathbf{x}) = \begin{cases} f_j(\mathbf{x}) - pen(\mathbf{x}), & j = 1,2,\ldots,k, \quad \text{se } P[i] \text{ é inviável} \\ f_j(\mathbf{x}), & j = 1,2,\ldots,k, \quad \text{se } P[i] \text{ é viável.} \end{cases} \tag{4.5}$$

4.1.4 Seleção

Após a avaliação dos ajustes de todos os indivíduos da população, inicia-se a etapa de seleção. Primeiramente, é avaliada a relação de dominância entre as soluções com respeito às funções $f_1^{\text{pen}}, \ldots, f_k^{\text{pen}}$. Os indivíduos dominados são eliminados e os não-dominados prosseguem em P. Uma cópia de cada indivíduo não-dominado é inserida em P_{aux}. Na primeira geração, P_{aux} é igual a P.

Como os indivíduos dominados são eliminados, o tamanho de P é reduzido para N_r ($N_r < N$). Para que N continue sendo o número de indivíduos em P, sorteiam-se aleatoriamente $N - N_r$ indivíduos de P_{aux} e esses são acrescentados em P.

A população P_{aux} recebe indivíduos não-dominados *no contexto da população corrente*, sem avaliar a relação de dominância entre eles e as soluções nela já existentes. Assim indivíduos não-dominados de uma geração podem ser dominados por soluções não-dominadas de uma outra geração, o que implica que P_{aux} pode conter indivíduos dominados. A atualização da relação de dominância entre os indivíduos de P_{aux} *não* é feita a cada iteração com o intuito introduzir uma certa variabilidade às soluções. A diversidade dos indivíduos é um fator importante para a exploração do espaço de busca.

4.1.5 *Crossover*

Após a seleção, sorteiam-se números aleatórios para cada indivíduo de P. Se esse número for menor ou igual à probabilidade de *crossover* p_c, então o indivíduo participará do *crossover*.

Utilizou-se o *crossover* de q-pontos, onde q indica o número de pontos de corte em cada par de indivíduos pais. A escolha das posições dos cortes no genótipo dos pais se dá por meio de números aleatórios gerados no intervalo $[1, tam - 1]$, onde tam representa o tamanho do indivíduo (a quantidade de posições existentes em seu genótipo). Na figura 4.3 é ilustrado um *crossover* de 3-pontos para indivíduos com $tam = 12$ e as posições selecionadas para corte foram 2, 5 e 9. Os pais têm seus materiais genéticos trocados alternadamente de acordo com os cortes estabelecidos e assim geram os filhos 1 e 2. O filho 1 tem material genético do pai 2 nas duas primeiras posições e nas posições de 6 a 9 e material genético do pai 1 nas demais posições. Já no filho 2, os materiais genéticos dos pais 1 e 2 são alocados de forma oposta à alocação realizada para o filho 1.

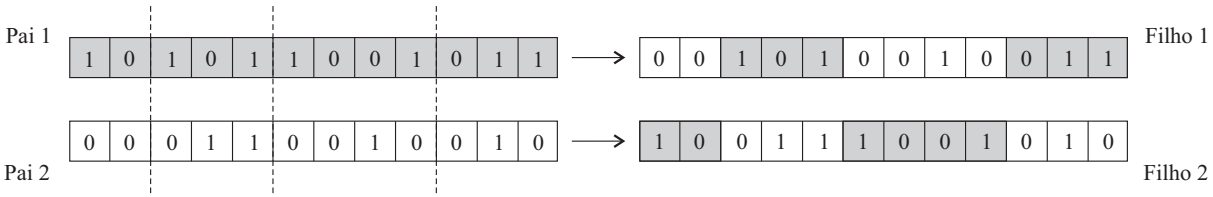

Figura 4.3: *Crossover* de 3-pontos

4.1.6 Substituição

Adotou-se a estratégia de substituição em que os pais cedem lugar aos filhos. Quando esse tipo de substituição é usado, após o *crossover*, os pais podem ser imediatamente eliminados de P e seus respectivos filhos tomam suas posições. Nesse caso, a mutação é aplicada uma única vez (apenas em P) e após a etapa de substituição. Porém, se a população da próxima geração for formada com os melhores indivíduos, a mutação deve ser aplicada tanto em P quanto nos indivíduos resultantes do *crossover* (duas vezes) e antes da etapa de substituição.

4.1.7 Mutação

A mutação consiste em trocar o conteúdo de uma posição do genótipo do indivíduo. Ou seja, se ele for 0, passa a ser 1 e vice-versa. Sorteia-se um número aleatório para cada posição do indivíduo. Se esse número aleatório for menor ou igual que a probabilidade de mutação p_m, troca-se o valor da posição, como é mostrado na figura 4.4. Esse procedimento se repete para toda a população e é chamado de Mutação.

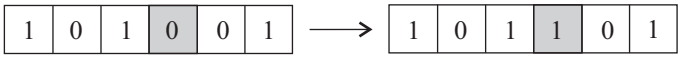

Figura 4.4: Mutação

Depois da mutação, completa-se uma iteração e o algoritmo é repetido N_{ger} vezes, onde N_{ger} é o número de gerações. Ao final de todas as iterações, o algoritmo retorna os indivíduos de P_{aux} que sejam *viáveis* e *não-dominados com respeito às funções-objetivo*.

4.2 Pseudocódigo do Algoritmo Genético Multiobjetivo Proposto

A seguir é apresentado o algoritmo genético resultante de tudo que foi explicado e discutido neste capítulo. Ele é apresentado em forma de pseudocódigo. O símbolo ▷ indica o início de comentários. Com o que foi visto e com os comentários no pseudocódigo, deve ficar clara a essência do método proposto.

procedure ALGORITMOGENÉTICO

$$\left(\begin{array}{rl} N,\, N_{\text{ger}} & \triangleright \text{ tamanho da população e total de gerações} \\ \ell_1,\, u_1,\, \ell_2,\, u_2,\, \ldots,\, \ell_n,\, u_n, & \triangleright \text{ limites das } n \text{ variáveis de decisão} \\ f_1,\, \ldots,\, f_k, & \triangleright \text{ funções-objetivo} \\ g_1,\, \ldots,\, g_p, & \triangleright \text{ restrições de igualdade} \\ h_{p+1},\, \ldots,\, h_m, & \triangleright \text{ restrições de desigualdade} \\ p_c,\, p_m & \triangleright \text{ probabilidades de } \textit{crossover} \text{ e de } \textit{mutação} \end{array}\right)$$

$P \leftarrow \langle\,\rangle, P_{\text{aux}} \leftarrow \langle\,\rangle$ ▷ inicializar vetores de indivíduos P e P_{aux} como vazios (*i.e.* comprimento zero)

for $j = 1 \ldots N$ **do** ▷ concatenar N indivíduos gerados aleatoriamente ao vetor P

$P \leftarrow P \cdot \langle \text{INDRAND}(\ell_1, u_1, \ell_2, u_2, \ldots, \ell_n, u_n) \rangle$

for $g = 1 \ldots N_{\text{ger}}$ **do** ▷ laço das gerações

▷ inicializar vetores J_B e J_R com índices de indivíduos de P, respectivamente, *não-dominados* e *dominados* em P com respeito às funções $f_1^{\text{pen}} \ldots f_k^{\text{pen}}$; concatenar ao vetor P_{aux} os indivíduos *não-dominados* de P

$J_B \leftarrow \langle\,\rangle$, $J_R \leftarrow \langle\,\rangle$

for $j = 1 \ldots N$ **do**

if $P[j]$ é *não-dominado* em P com respeito às funções $f_1^{\text{pen}} \ldots f_k^{\text{pen}}$ **then**

$J_B \leftarrow J_B \cdot \langle j \rangle$

$P_{\text{aux}} \leftarrow J_B \cdot \langle P[j] \rangle$

else

$J_R \leftarrow J_R \cdot \langle j \rangle$

▷ substituir indivíduos em P com índice em J_R por indivídos sorteado aleatoriamente em P_{aux}

for $j \in J_R$ **do**

$P[j] \leftarrow P_{\text{aux}}[\text{RAND}(\{1, 2, \ldots, |P_{\text{aux}}|\})]$

▷ escolher indivíduos de P para *crossover* com probabilidade p_c, estes índividuos são guardados no vetor de índices J_C

$J_C \leftarrow \langle\,\rangle$

for $j = 1 \ldots N$ **do**

if $\text{RAND}([0,1]) < p_c$ **then**

$J_C \leftarrow J_C \cdot \langle j \rangle$

▷ substituir em P o par de indivíduos nas posições $J_C[j]$ e $J_C[j+1]$ pelo resultado de um *crossover* neste mesmo par de indivíduos

$j \leftarrow 1$

while $j + 1 < |J_C|$ **do**

$P[J_C[j]], P[J_C[j+1]] \leftarrow \text{CROSSOVER}(P[J_C[j]], P[J_C[j+1]])$

$j \leftarrow j + 2$

▷ realizar *mutação* nos indivíduos de P com probabilidade p_m

for $j = 1 \ldots N$ **do**

if $\text{RAND}([0,1]) < p_m$ **then**

$P[j] \leftarrow \text{MUTATION}(P[j])$

end for ▷ fim do laço das gerações

return indivíduos viáveis de P_{aux} que são *não-dominados* em P_{aux} com respeito às funções $f_1 \ldots f_k$

end procedure ▷ fim do procedimento

5 EXEMPLOS DE APLICAÇÃO

Nesse capítulo são considerados dois exemplos de aplicação da abordagem multiobjetivo proposta para o problema de alocação de redundâncias.

No primeiro exemplo, buscam-se as configurações de um sistema não-reparável paralelo-série que maximizam sua confiabilidade e minimizam seu custo. Os dois objetivos podem ser calculados analiticamente.

O segundo exemplo é mais complexo e deseja-se encontrar as configurações de um sistema reparável paralelo-série que maximizem sua disponibilidade e minimizem o seu custo. Os componentes que formam o sistema têm processos de falha-reparo regidos por processos de renovação alternados, de modo que a superposição de tais processos caracteriza o processo de falha-reparo do sistema. Assim, a disponibilidade foi obtida diretamente da simulação discreta de eventos. O cálculo do custo de manutenções corretivas também depende da simulação, uma vez que utiliza o número de reparos médios por componente ao longo do tempo de missão. O custo de aquisição pode ser obtido por meio de uma expressão analítica.

5.1 Exemplo 1

O exemplo 1 é uma adaptação do exemplo apresentado em (TABOADA; COIT, 2006). O problema considera um sistema paralelo-série formado por três subsistemas S_1, S_2 e S_3 (figura 5.1). Em cada subsistema deve haver no mínimo um componente e pode ter no máximo oito componentes em paralelo. Considera-se ainda que S_1 e S_3 têm, cada um, cinco opções de componentes que exercem a mesma função, mas possuem custos e confiabilidades diferentes (na prática, os componentes podem ser produzidos por fabricantes diferentes). Já S_1 tem quatro opções de componentes.

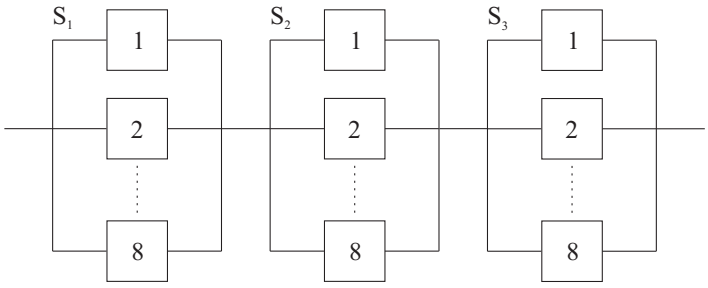

Figura 5.1: Configuração do sistema do exemplo 1

A formulação matemática geral desse problema é dada a seguir:

$$\text{Maximize} \quad \prod_{i=1}^{s} R_i(\mathbf{x_i}) \tag{5.1}$$

$$\text{Minimize} \quad \sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij} \tag{5.2}$$

$$\begin{aligned} \text{Sujeito a} \quad & \sum_{j=1}^{m_i} x_{ij} \leq n_i^{\max}, \quad i = 1,2,\ldots,s \\ & \sum_{j=1}^{m_i} x_{ij} \geq 1, \quad i = 1,2,\ldots,s \\ & x_{ij} \in \mathbb{Z}^+, \end{aligned}$$

onde s é o número de subsistemas, x_{ij} é a quantidade de componentes do tipo j usados no subsistema i, m_i é o número total de componentes no subsistema i, $n_i^{\max}$ é o número máximo de componentes em paralelo que podem ser usados no subsistema i, $\mathbf{x_i}$ é igual a $\{x_{i1}, x_{i2}, \ldots, x_{im_i}\}$, $R_i(\mathbf{x_i})$ é a confiabilidade do subsistema i e c_{ij} é o custo de aquisição do j-ésimo componente usado no subsistema i.

A confiabilidade R_i do subsistema i é dada por

$$R_i = 1 - \prod_{j=1}^{m_i}(1 - r_{ij}), \tag{5.3}$$

em que r_{ij} é a confiabilidade do j-ésimo componente usado no subsistema i.

Para o exemplo de aplicação $s = 3$, $n_i^{\max} = 8, i = 1,2,3$ e as confiabilidades e os custos de aquisição de cada tipo de componente (por subsistema) são mostrados na tabela 5.1.

Tabela 5.1: Confiabilidade (r_{ij}) e custo de aquisição (c_{ij}) dos componentes. Fonte: (TABOADA; COIT, 2006, p. 24, adaptação)

	S_1		S_2		S_3	
Componente	r_{ij}	c_{ij}	r_{ij}	c_{ij}	r_{ij}	c_{ij}
1	0,94	900	0,97	1200	0,96	1000
2	0,91	600	0,86	300	0,89	600
3	0,89	600	0,70	200	0,72	400
4	0,75	300	0,66	200	0,71	300
5	0,72	200	–	–	0,67	200

Para solucionar esse exemplo de aplicação, utilizou-se o algoritmo multiobjetivo desenvolvido nesse trabalho. Um exemplo de fenótipo associado aos indivíduos é mostrado na figura 5.2, que é formado por catorze variáveis de decisão: as cinco primeiras referentes a S_1, as outras

quatro relacionadas a S_2 e as cinco últimas associadas a S_3.

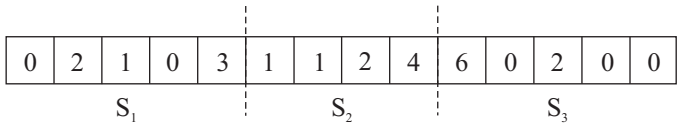

Figura 5.2: Exemplo de fenótipo para indivíduos do exemplo 1

Os diversos parâmetros usados no algoritmo genético multiobjetivo são definidos na tabela 5.2.

Tabela 5.2: Parâmetros usados no algoritmo genético multiobjetivo para o exemplo 1

Parâmetro	**Valor**
Tamanho da população	100
Número de gerações	200
Probabilidade de *crossover*	0,95
Número de pontos de corte para *crossover*	5
Probabilidade de mutação	0,01

Após executar o algoritmo para todas as gerações, a população auxiliar P_{aux}, dentre todos os seus indivíduos, possuía 3647 indivíduos *viáveis*, que foram a ela adicionados porque em alguma iteração eles foram não-dominados *no contexto da população corrente*. O gráfico da figura 5.3 mostra esses pontos viáveis. Quando se avaliou a relação de dominância entre esses indivíduos, restaram 67 soluções não-dominadas, que estão em destaque (cor preta) na figura 5.3 e formam a fronteira apresentada no gráfico da figura 5.4.

Para ilustrar, foram selecionados algumas soluções da fronteira encontrada soluções (figura 5.4). As configurações representadas por essas soluções são mostradas na figura 5.5, onde os números em cada bloco representam o tipo de componente para cada subsistema, conforme a tabela 5.1. A solução A apresenta os menores valores tanto para a confiabilidade quanto para o custo. Já a solução C apresenta a maior confiabilidade e o maior custo dentre todas as soluções da fronteira. As soluções A e C são os extremos da fronteira encontrada. As soluções B, C, D e E têm desempenho entre as soluções A e C, já que apresentam uma confiabilidade mais alta que a da solução A e mais baixa que a da solução C e isso acontece de forma análoga para o custo. A tabela 5.3 mostra os valores dos objetivos para as soluções A, B, C, D, E e F.

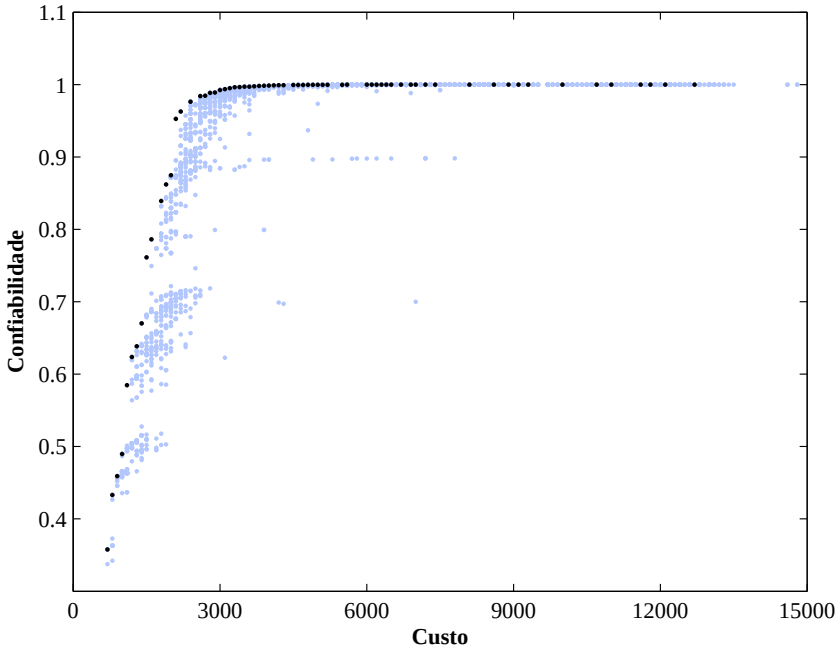

Figura 5.3: Pontos viáveis e fronteira de soluções não-dominadas para o exemplo 1

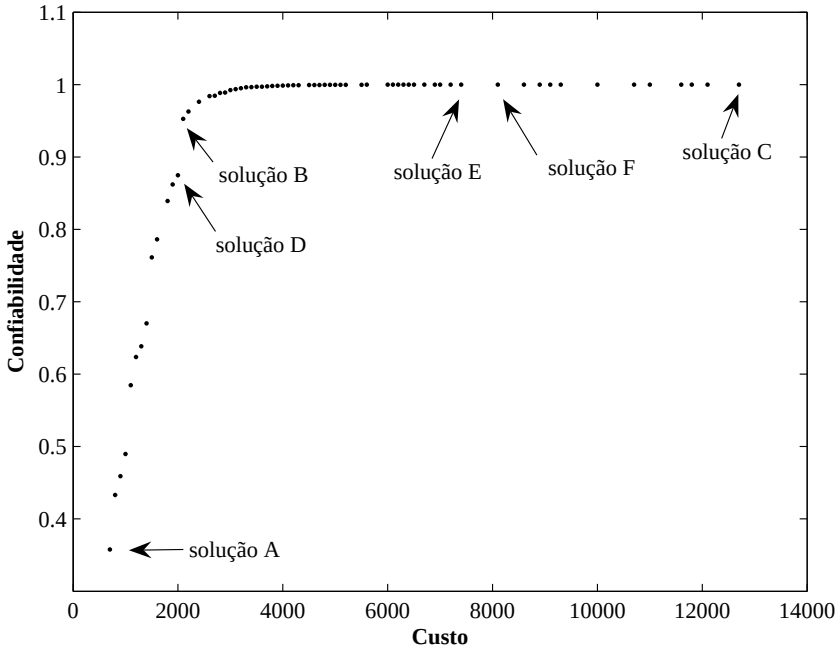

Figura 5.4: Fronteira de soluções não-dominadas para o exemplo 1

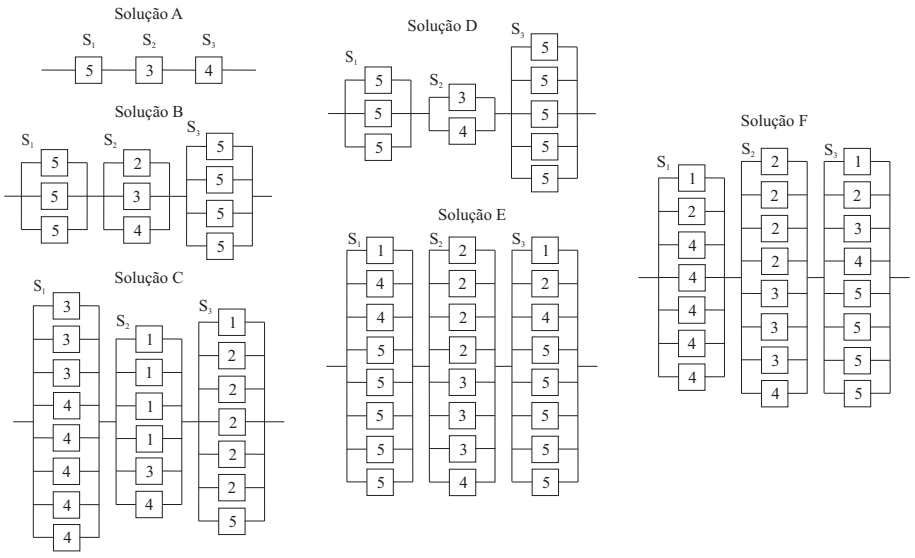

Figura 5.5: Soluções selecionadas da fronteira de indivíduos não-dominados, exemplo 1

Tabela 5.3: Exemplos de soluções para o problema do exemplo 1

Solução	Confiabilidade	Custo
A	0,357840	700
B	0,952648	2100
C	0,999998	12700
D	0,874850	2000
E	0,999985	7400
F	0,999987	8100

Análise do retorno sobre investimento

Do ponto de vista multiobjetivo todas as soluções da fronteira no gráfico da figura 5.4 são ótimas. Uma sugestão para avaliar as soluções não-dominadas encontradas consiste em analisar o ganho de confiabilidade em relação ao investimento feito no projeto do sistema:

$$\text{Retorno } = \frac{R_i - R_{i-1}}{C_i - C_{i-1}}, \tag{5.4}$$

em que as soluções $i-1$ e i são pontos adjacentes da fronteira de Pareto e R_{i-1}, R_i, C_{i-1} e C_i são suas respectivas confiabilidades e custos. É interessante que o numerador da equação (5.4) tenha um valor grande e que a diferença $C_i - C_{i-1}$ seja pequena, para que se ontenha um alto retorno.

Para ilustrar, considere as soluções D e B e as soluções E e F (figura 5.4). A tabela 5.4 mostra os valores dos objetivos para cada solução e os retornos, obtidos da equação (5.4).

Ao se analisar a tabela 5.4, pode ser observado que partindo-se da solução D, acrescenta-se 0,077798 à confiabilidade do sistema investindo-se 100 unidades monetárias. Partindo-se

Tabela 5.4: Retorno sobre investimento de algumas soluções do exemplo 1

Solução	**Confiabilidade**	**Custo**	**Retorno**
D	0,874850	2000	0,000778
B	0,952648	2100	
E	0,999985	7400	$2 \cdot 10^{-9}$
F	0,999987	8100	

da solução E e investindo-se 700 unidades monetárias, o sistema terá um aumento de apenas 0,000002 na confiabilidade, o que na prática pode não ser relevante.

Note que se esse exemplo fosse resolvido de forma exata, isto é, testando-se todas as possibilidades e se verificando a relação de dominância entre elas, seria necessário avaliar 816.975.224 combinações.

5.2 Exemplo 2

Nesse exemplo, considera-se um sistema paralelo-série formado de dois subsistemas S_1 e S_2 constituídos de componentes reparáveis. S_1 pode ter no máximo dois componentes em paralelo, enquanto S_2 pode ter no máximo quatro e ambos devem ter no mínimo um componente (figura 5.6). Além disso, S_1 e S_2 têm respectivamente cinco e seis possibilidades de componentes que exercem a mesma função, mas possuem processos de falha, custos de aquisição e de manutenção corretiva diferentes. Um exemplo de fenótipo para um indivíduo modelado para esse exemplo é mostrado na figura 5.7.

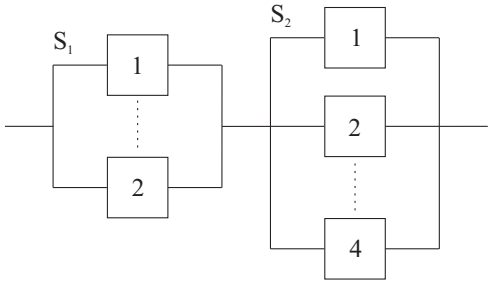

Figura 5.6: Configuração do sistema do exemplo 2

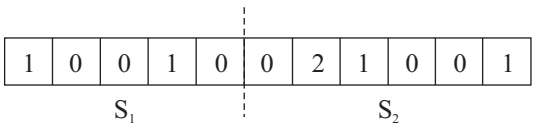

Figura 5.7: Exemplo de fenótipo para indivíduos do exemplo 2

Assume-se que cada um dos componentes segue um processo de renovação alternado, ou seja, após a falha são submetidos ao reparo perfeito e conseqüentemente voltam à operação numa condição de "tão bons quanto novos". O sistema tem, portanto, uma superposição de processos de renovação alternados e o tratamento analítico dessa superposição é inviável. Assim, optou-se pela simulação discreta de eventos. Utilizou-se um simulador de eventos discretos desenvolvido pelo grupo RISCTEC-UFPE para simular o comportamento do sistema no tempo de missão estabelecido. Como os dois objetivos (disponibilidade e custo) dependem diretamente do simulador, foi necessária a comunicação entre o simulador e o algoritmo genético multiobjetivo proposto.

A disponibilidade do sistema foi obtida a partir da simulação do comportamento do sistema para o tempo de missão estabelecido e calculada conforme a expressão

$$A_S = \frac{\text{Tempo operacional}}{\text{Tempo total}}. \tag{5.5}$$

O custo associado ao sistema é função do custo de aquisição dos componentes e do custo de manutenção corretiva dos mesmos. Assim, a função custo que se deseja minimizar é dada por

$$C_S = C_A + C_{MC}, \tag{5.6}$$

em que C_A é o custo total de aquisição dos componentes obtido da expressão

$$C_A = \sum_{i=1}^{s} \sum_{j=1}^{m_i} c_{ij}^{a} x_{ij}, \tag{5.7}$$

onde c_{ij}^{a} é o custo de adquirir uma unidade do j-ésimo componente do subsistema i. Já C_{MC} é o custo total de manutenção corretiva dos componentes ao longo do tempo de missão:

$$C_{CM} = \sum_{i=1}^{s} \sum_{j=1}^{m_i} \sum_{k=1}^{x_{ij}} c_{ij}^{mc} n_{ijk} \tag{5.8}$$

onde n_{ijk} é o número médio de reparos da k-ésima cópia do j-ésimo tipo de componente do subsistema i e c_{ij}^{mc} é o custo de reparar uma unidade do j-ésimo tipo de componente do subsistema i. Assim como a disponibilidade do sistema, número médio de manutenções corretivas realizadas em cada componente durante o tempo de missão é obtido da simulação.

O custo total C_S associado ao sistema é portanto

$$C_S = \sum_{i=1}^{s} \sum_{j=1}^{m_i} \left[c_{ij}^{a} x_{ij} + \sum_{k=1}^{x_{ij}} c_{ij}^{mc} n_{ijk} \right] \tag{5.9}$$

A tabela 5.5 mostra as distribuições dos tempos entre falhas (distribuição Weibull com parâmetros de escala α, em unidade de tempo, e de forma β), as distribuições dos tempos de reparo (distribuição exponencial com parâmetro λ), o custo unitário de aquisição e o custo unitário de manutenção corretiva dos cinco tipos de componentes de S_1 e dos seis tipos de componentes de S_2.

Tabela 5.5: Características dos componentes considerados no exemplo 2

	S_1				S_2			
Componente	$f_X(x)$	$f_D(d)$	c^a	c^{mc}	$f_X(x)$	$f_D(d)$	c^a	c^{mc}
1	$Weibull(20;1,5)$	$Exp(0,2)$	590	30	$Weibull(20;1,2)$	$Exp(0,2)$	670	60
2	$Weibull(25;1,5)$	$Exp(0,2)$	730	55	$Weibull(30;2,0)$	$Exp(0,2)$	500	20
3	$Weibull(27;1,7)$	$Exp(0,2)$	650	45	$Weibull(40;1,8)$	$Exp(0,2)$	1100	90
4	$Weibull(40;1,4)$	$Exp(0,2)$	1000	85	$Weibull(37;1,5)$	$Exp(0,2)$	1340	110
5	$Weibull(35;1,2)$	$Exp(0,2)$	850	78	$Weibull(32;1,6)$	$Exp(0,2)$	1215	95
6	–	–	–	–	$Weibull(29;1,3)$	$Exp(0,2)$	950	90

O tempo de missão utilizado na simulação do comportamento do sistema foi de 100 unidades de tempo. Os diversos parâmetros usados no algoritmo genético multiobjetivo são definidos na tabela 5.6.

Tabela 5.6: Parâmetros usados no algoritmo genético multiobjetivo para o exemplo 2

Parâmetro	**Valor**
Tamanho da população	50
Número de gerações	100
Probabilidade de *crossover*	0,95
Número de pontos de corte para *crossover*	4
Probabilidade de mutação	0,01

Após executar o algoritmo, foram encontrados 1369 indivíduos viáveis, que em algum momento do processo evolutivo foram não-dominados *no contexto da população corrente* (figura 5.8). Desses, 54 soluções foram globalmente não-dominadas e são mostradas em cor preta no gráfico da figura 5.8 e na fronteira apresentada na figura 5.9.

Assim como no exemplo 1, foram selecionadas soluções para ilustrar algumas das configurações encontradas (figura 5.10, os números em cada bloco representam o tipo de componente para o subsistema). As soluções A e C são os pontos extremos da fronteira e as soluções B,D e E apresentam desempenho intermediário em ambos os objetivos. A tabela 5.7 mostra os valores dos objetivos para as soluções A, B, C, D e E.

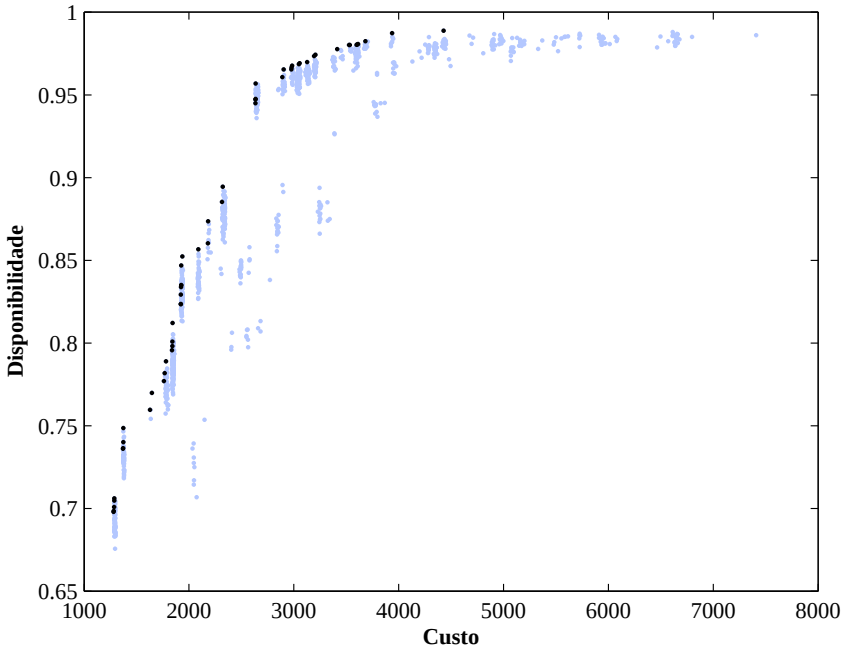

Figura 5.8: Pontos viáveis e fronteira de soluções não-dominadas para o exemplo 2

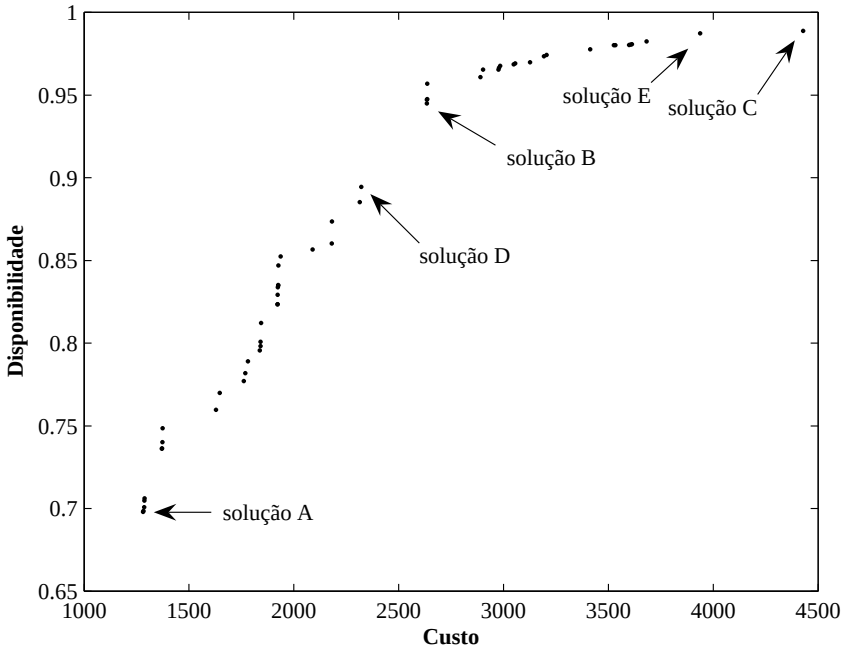

Figura 5.9: Fronteira de soluções não-dominadas para o exemplo 2

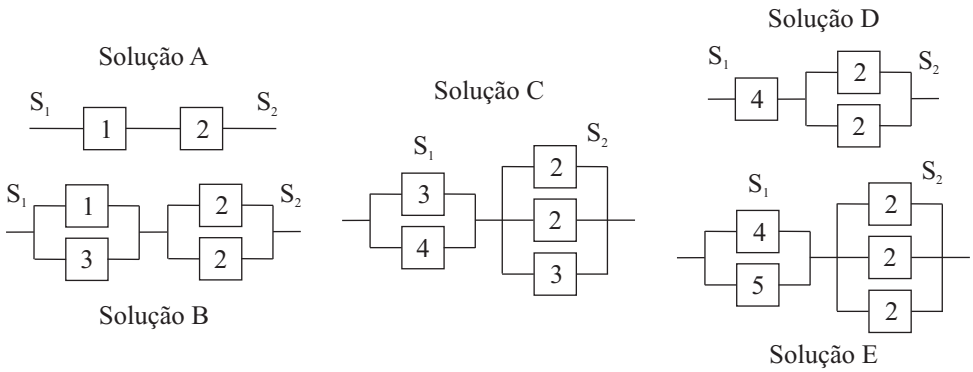

Figura 5.10: Soluções selecionadas da fronteira de indivíduos não-dominados, exemplo 2

Tabela 5.7: Exemplos de soluções para o problema do exemplo 2

Solução	**Disponibilidade**	**Custo**
A	0,700909	1286,40
B	0,944968	2634,35
C	0,988761	4428,3
D	0,894504	2321,80
E	0,987361	3937,97

Análise do retorno sobre investimento

Da mesma maneira que no caso do sistema não-reparável do exemplo 1, pode-se analisar as soluções da fronteira obtida para esse exemplo. No entanto, considera-se a disponibilidade em vez da confiabilidade. Assim, a equação (5.4) é modificada para:

$$\text{Retorno } = \frac{A_i - A_{i-1}}{C_i - C_{i-1}}, \tag{5.10}$$

em que as soluções $i-1$ e i são pontos adjacentes da fronteira de Pareto e A_{i-1}, A_i, C_{i-1} e C_i são suas respectivas disponibilidades e custos.

Selecionando-se as soluções D e B e E e C (figura 5.9), obtêm-se os valores dos retornos, apresentados na tabela 5.8, aplicando-se a equação (5.10).

Tabela 5.8: Retorno sobre investimento de algumas soluções do exemplo 2

Solução	**Disponibilidade**	**Custo**	**Retorno**
D	0,894504	2321,80	0,000161
B	0,944968	2634,35	
E	0,987361	3937,97	0,000003
C	0,988761	4428,30	

Por meio da análise da tabela 5.8, observa-se que para o sistema ter uma disponibilidade de 0,944968 em vez de 0,894504, é preciso investir 312,55 unidades monetárias. Já para a dispo-

nibilidade do sistema ser igual a 0,988761 em vez de 0,987361, adiciona-se 490,33 unidades monetárias ao custo total do sistema. Nesse último caso o ganho de disponibilidade do sistema é muito pequeno se comparado ao investimento que se deve fazer.

Nos dois exemplos apresentados, além da análise do retorno sobre o investimento, pode-se utilizar métodos multiatributo de apoio a decisão (VINCKE, 1992; YOON; HWANG, 1995; ROY, 1996; GOMES *et al.*, 2002) para auxiliar o decisor na escolha de uma das soluções não-dominadas encontradas.

6 CONSIDERAÇÕES FINAIS

6.1 Conclusões

A elaboração desse trabalho possibilitou um conhecimento mais aprofundado dos temas abordados: projetos de sistemas baseados em confiabilidade, sistemas reparáveis, sistemas não-reparáveis, processos de renovação, simulação discreta de eventos, otimização multiobjetivo, algoritmos evolucionários, além da linguagem de programação C++, utilizada na implementação do algoritmo genético multiobjetivo.

Nesse trabalho buscou-se utilizar uma abordagem multiobjetivo para encontrar configurações de sistemas que apresentem soluções satisfatórias para os objetivos considerados – métricas de desempenho (confiabilidade ou disponibiliade) e custos associados. Para isso, desenvolveu-se um algoritmo evolucionário – algoritmo genético – em que a etapa de seleção utiliza os conceitos de dominância e não-dominância, o que permitiu o tratamento multiobjetivo dos problemas relacionados a projetos de sistemas.

Foram resolvidos dois exemplos de aplicação do problema de alocação de redundâncias em sistemas paralelo-série. Um considerando um sistema não-reparável em que se procurou otimizar a confiabilidade e o custo de aquisição de componentes e o outro levando em conta um sistema reparável em que se buscou maximizar a disponibilidade do sistema e minimizar os custos associados (custo de aquisição e de manutenção corretiva dos componentes). No segundo exemplo, assumiu-se que cada um dos componentes era submetido ao reparo perfeito, e o tempo de tal reparo *não* foi desconsiderado, supôs-se que cada componente seguia um processo de renovação alternado.

A disponibilidade e o número médio de reparos (usado no cálculo do custo) dos componentes do sistema reparável, avaliado no segundo exemplo, foram obtidos por meio de simulação, uma vez que ele considerou suposições mais realistas e uma abordagem analítica tornou-se inviável.

Para avaliar as soluções não-dominadas encontradas, sugeriu-se uma análise do ganho de confiabilidade (ou disponibilidade) em relação ao investimento que se deve fazer no projeto do sistema.

6.2 Limitações

A principal desvantagem dos métodos evolucionários de otimização é o fato de eles não garantirem o ótimo. Na abordagem multiobjetivo, encontra-se um conjunto de soluções não-dominadas que não necessariamente pertencem à fronteira real de Pareto.

Em relação ao algoritmo desenvolvido, foi usada a codificação binária, que interfere na

velocidade do mesmo, deixando-o um pouco mais lento.

Com respeito aos exemplos apresentados, apesar de o segundo ser mais realista que o primeiro, a hipótese de reparos perfeitos é muito forte e não é comum na prática. Além disso, a função custo considerou apenas o custo de aquisição (no exemplo 1) e o custo de aquisição e de manutenção corretiva (no exemplo 2).

6.3 Sugestões para Trabalhos Futuros

Algumas das sugestões para trabalhos futuros são:

- Avaliar sistemas reparáveis sujeitos a reparos imperfeitos, por meio de processos de renovação generalizados.
- Incorporar outras políticas de manutenção à modelagem do sistema, como a preventiva, e avaliar os impactos de tais políticas no desempenho e nos custos associados ao sistema.
- Considerar mais objetivos a serem otimizados. Além de maximizar disponibilidade / confiabilidade e minimizar custos, pode-se, por exemplo, minimizar o peso total dos componentes do sistema.
- Considerar o custo de operação de cada componente.
- Utilizar outros tipos de codificação / decodificação, operadores genéticos e substituição.
- Comparar os impactos na variabilidade pelas estratégias de penalizar indivíduos e de *não* gerar indivíduos inviáveis.
- Otimizar confiabilidade / disponibilidade e custo para outras configurações de sistemas, como a série-paralelo.
- Comparar o desempenho do algoritmo desenvolvido com outros algoritmos genéticos multiobjetivo.

REFERÊNCIAS BIBLIOGRÁFICAS

ARENALES, M.; ARMENTANO, V. A.; MORABITO, R.; YANASSE, H. H. *Pesquisa operacional*. Rio de Janeiro: Elsevier, 2007.

BÄCK, T.; HAMMEL, U.; SCHWEFEL, H.-P. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, vol. 1, n. 1, p. 3–17, 1997.

BANKS, J.; CARSON, J. S.; NELSON, B. L.; NICOL, D. M. *Discrete event system simulation*. 3ed. Upper Saddle River: Prentice Hall, 2001.

BEYER, H.-G.; SCHWEFEL, H.-P. Evolution strategies: a comprehensive introduction. *Neural Computing*, Kluwer Academic, vol. 1, p. 3–52, 2002.

BOLDRINI, J. L.; COSTA, S. I. R.; FIGUEIREDO, V. L.; WETZLER, H. G. *Álgebra Linear*. 3ed. São Paulo: Harper & Row do Brasil, 1980.

BUSACCA, P. G.; MARSEGUERRA, M.; ZIO, E. Multiobjective optimization by genetic algorithms: application to safety systems. *Reliability Engineering & System Safety*, vol. 72, p. 59–74, 2001.

COELLO, C. A. C. An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends. In: *Congress on Evolutionary Computation*. IEEE, 1999. v. 1, p. 3–13.

COELLO, C. A. C.; VELDHUIZEN, D. A. V.; LAMONT, G. B. *Evolutionary algorithms for solving multiobjective problems*. New York: Kluwer Academic, 2002.

DAS, I.; DENNIS, J. E. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural and Multidisciplinary Optimization*, vol. 14, n. 1, p. 63–69, 1997.

DEB, K. Evolutionary algorithms for multi-criterion optimization in engineering design. In: *Evolutionary Algorithms in Engineering and Computer Science (EUROGEN'99)*. 1999.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, vol. 6, n. 2, p. 182–197, 2002.

DHINGRA, A. K. Optimal apportionment of reliability & redundancy in series systems under multiple objectives. *IEEE Transactions on Reliability*, vol. 41, n. 4, p. 576–583, 1992.

FISHMAN, G. S. *Monte Carlo: concepts, algorithms and applications*. Springer, 2000.

FOGEL, L. J.; ANGELINE, P. J.; FOGEL, D. B. An evolutionary programming approach to self-adaptation on finite state machines. In: *Proceedings of the Fourth Annual Conference on Evolutionary Programming*. 1995.

FONSECA, C. M.; FLEMING, P. J. Genetic algorithms for multi-objective optimization: formulation, discussion and generalization. In: *Proceedings of the Fifth International Conference on Genetic Algorithms*. 1993.

GOMES, L. F. A. M.; GOMES, C. F. S.; ALMEIDA, A. T. de. *Tomada de decisão gerencial: enfoque multicritério*. São Paulo: Atlas, 2002.

HERRERA, F.; LOZANO, M.; VERDEGAY, J. L. Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. *Artificial Intelligence Review*, vol. 12, p. 265–319, 1998.

HILLIER, F. S.; LIEBERMAN, G. J. *Introduction to operations research*. San Francisco: Holden-Day, 1967.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. E. A niched pareto genetic algorithm for multiobjective optimization. In: *Proceedings of the First IEEE Conference on Evolutionary Computation*. 1994.

JOINES, J. A.; HOUCK, C. R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with gas. In: *Proceedings of the First IEEE International Conference on Evolutionary Computation*. 1994.

KIRKPATRICK, S.; JR., C. D. G.; VECCHI, M. P. Optimization by simulated annealing. *Science*, vol. 220, p. 671–680, 1983.

KUO, W.; PRASAD, V. R.; TILLMAN, F. A.; HWANG, C.-L. *Optimal reliability design: fundamentals and applications*. United Kingdom: Cambridge University Press, 2001.

LAPA, C. M. F.; PEREIRA, C. M. N. A.; BARROS, M. P. de. A model for preventive maintenance planning by genetic algorithms based in cost and reliability. *Reliability Engineering & System Safety*, vol. 91, p. 233–240, 2006.

LUENBERGER, D. G. *Linear and nonlinear programming*. 2ed. Massachusetts: Addison-Wesley, 1984.

MARSEGUERRA, M.; ZIO, E.; PODOFILLINI, L. Multiobjective spare part allocation by means of genetic algorithms and monte carlo simulation. *Reliability Engineering & System Safety*, vol. 87, p. 325–335, 2005.

MARTORELL, S.; CARLOS, S.; SÁNCHEZ, A.; SERRADELL, V. Constrained optimization of test intervals using steady-state genetic algorithm. *Reliability Engineering & System Safety*, vol. 67, p. 215–232, 2000.

MESSAC, A.; SUNDARARAJ, G. J.; TAPPETA, R. V.; RENAUD, J. E. Ability of objective functions to generate points on nonconvex pareto frontiers. *AIAA Journal*, vol. 38, n. 6, p. 1084–1091, 2000.

MODARRES, M. *Risk analysis in engineering: techniques, tools and trends*. Boca Raton: Taylor & Francis, 2006.

RAUSAND, M.; HOYLAND, A. *System reliability theory: models and statistical methods*. 2ed. New York: John Wiley & Sons, 2003.

RIGDON, S. E.; BASU, A. P. *Statistical methods for the reliability of repairable systems*. New York: John Wiley & Sons, 2000.

ROSS, S. M. *Introduction to probability models*. 7ed. San Diego: Academic Press, 2000.

ROSS, S. M. *Simulation*. 3ed. San Diego: Academic Press, 2002.

ROY, B. *Multicriteria methodology for decision aiding*. Kluwer Academic, 1996.

SCHAFFER, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In: *Proceedings of the First International Conference on Genetic Algorithms*. 1985.

SRINIVAS, N. K.; DEB, K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Journal of Evolutionary Computation*, vol. 2, n. 3, p. 221–248, 1994.

TABOADA, H. A.; COIT, D. W. MOEA-DAP: a new multiple objective evolutionary algorithm for solving design allocation problems. *Reliability Engineering & System Safety (under review)*, 2006. Disponível em: http://www.engr.rutgers.edu/ ie/research/working_paper/paper%2006-026.pdf. Acesso em 13/07/2007.

TABOADA, H. A.; COIT, D. W. Multiple objective design allocation problems: development of new evolutionary algorithms. In: *Proceedings of the European Safety & Reliability Conference (ESREL)*. Stavanger, Norway: 2007.

TABOADA, H. A.; ESPIRITU, J.; COIT, D. W. MOMS-GA: a multiobjective multi-state genetic algorithm for system reliability optimization design problems. *IEEE Transactions on Reliability (in print)*, 2007.

TAPPETA, R. V.; RENAUD, J. E. Interactive multiobjective optimization design strategy for decision based design. *Journal of Mechanical Design*, vol. 123, p. 205–215, 2001.

VINCKE, P. *Multicriteria decision-aid*. Chichester: John Wiley & Sons, 1992.

WOLSEY, L. A. *Integer programming*. New York: John Wiley & Sons, 1998.

YALAOUI, A.; CHU, C.; CHÂTELET, E. Reliability allocation problem in series-parallel system. *Reliability Engineering & System Safety*, vol. 90, p. 55–61, 2005.

YAÑES, M.; JOGLAR, F.; MODARRES, M. Generalized renewal process for analysis of repairable systems with limited failure experience. *Reliability Engineering & System Safety*, vol. 77, p. 167–180, 2002.

YOON, K. P.; HWANG, C.-L. *Multiple attribute decision making: an introduction*. Thousand Oaks: Sage, 1995.

ZITZLER, E. *Evolutionary algorithms for multiobjective optimization: methods and applications*. Tese (Doutorado) — Swiss Federal Institute of Technology Zurich, 1999.

ZITZLER, E.; THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, vol. 3, n. 4, p. 257–271, 1999.