Notas em Matemática Aplicada

ISSN 2175-3385
E-ISSN 2236-5915

Volume 99, 2024

Sociedade Brasileira de Matemática Aplicada e Computacional

2024

A Sociedade Brasileira de Matemática Aplicada e Computacional - SBMAC publica, desde as primeiras edições do evento, monografias dos cursos que são ministrados nos CNMAC.

Para a comemoração dos 25 anos da SBMAC, que ocorreu durante o XXVI CNMAC em 2003, foi criada a série **Notas em Matemática Aplicada** para publicar as monografias dos minicursos ministrados nos CNMAC, o que permaneceu até o XXXIII CNMAC em 2010.

A partir de 2011, a série passa a publicar, também, livros nas áreas de interesse da SBMAC. Os autores que submeterem textos à série Notas em Matemática Aplicada devem estar cientes de que poderão ser convidados a ministrarem minicursos nos eventos patrocinados pela SBMAC, em especial nos CNMAC, sobre assunto a que se refere o texto.

O livro deve ser preparado em **Latex, com as figuras em .eps, .pdf e etc.** e ter entre **80 e 150 páginas**. O texto deve ser redigido de forma clara, acompanhado de uma excelente revisão bibliográfica e de **exercícios de verificação de aprendizagem** ao final de cada capítulo. O idioma pode ser Português ou Espanhol.

Veja todos os títulos publicados nesta série na página
http://https://proceedings.science/notas-sbmac

Sociedade Brasileira de Matemática Aplicada e Computacional

2024

OTIMIZAÇÃO AVANÇADA: EVOLUÇÃO DIFERENCIAL MELHORADA EM PARALELO

Milena Almeida Leite Brandão
milenabrandao@ufu.br
José Laércio Doricio
jldoricio@ufu.br

Instituto Ciências Exatas e Naturais do Pontal
Universidade Federal de Uberlândia

Sezimária de Fátima Pereira Saramago
saramago@ufu.br

Faculdade de Matemática
Universidade Federal de Uberlândia

Sociedade Brasileira de Matemática Aplicada e Computacional

São Carlos - SP, Brasil
2024

Coordenação Editorial: Mateus Bernardes

Coordenação Editorial da Série: Sandra M. C. Malta

Editora: SBMAC

Capa: Matheus Botossi Trindade

Patrocínio: SBMAC

Catalogação elaborada pela Biblioteca do IBILCE/UNESP
Bibliotecária: Maria Luiza Fernandes Jardim Froner

Brandão, Milena A. L.
Otimização Avançada: Evolução Diferencial Melhorada em Paralelo - São Carlos, SP :
SBMAC, 2024, 108 p., 21.5 cm - (Notas em Matemática Aplicada; v. 99)

ISBN 978-65-86388-27-5 e-ISBN 978-65-86388-28-2

1. Evolução Diferencial Melhorada 2. Evolução Diferencial 3. Otimização 4. Programação I. Brandão, Milena A. L. II. Dorício, José L. IV. Título. V. Série

CDD - 51

Eu, mamãe Milena, a meu filho Theo.
Dedico

Agradecimentos

Querido Theo,

É com imensa gratidão que escrevo esses agradecimentos. Agradeço a você, meu amado filho, pela incrível oportunidade de viver esta experiência ao seu lado.

Ao longo desta jornada, espero que, à medida que você cresça, compreenda as vezes em que não pude estar totalmente presente, quando estava ocupada ou até mesmo indisposta.

Enquanto digito essas palavras às 21h22min de uma segunda-feira, você está aqui, sentado ao meu lado, colorindo desenhos em rascunhos meus de papel A4. Cada interrupção com um "mamãe" é um lembrete da alegria que você traz à minha vida. Sei que, por vezes, precisei interromper a escrita para atendê-lo, mas sinto-me privilegiada por poder trabalhar ao seu lado. É verdade que não rende como eu gostaria mas essa é a vida de uma mãe cientista e pesquisadora. Obrigada filho por existir!

Com amor, mamãe!

Agradecemos também imensamente à nossa família e amigos, cujo apoio e compreensão tornaram essa trajetória possível.

Conteúdo

Prefácio

É com grande satisfação que apresentamos este livro, "Otimização Avançada: Evolução Diferencial Melhorada em Paralelo". Esta obra constitui uma exploração na abordagem da otimização matemática baseada em algoritmos evolutivos, especificamente a Evolução Diferencial (ED). O foco central é aprimorar a eficiência e a escalabilidade dessa técnica por meio de modificações no seu algoritmo e de sua implementação em processamento paralelo.

Os algoritmos evolutivos (AEs) são, em geral, métodos de otimização de busca estocástica que possuem como base da sua formulação a teoria da evolução. Nas últimas décadas o grande interesse dos pesquisadores por estes algoritmos tem impulsionado o desenvolvimento dos estudos, o que tem levado a uma significativa melhora na eficiência e aplicabilidade destes métodos para resolver problemas complexos de otimização nas diferentes áreas do conhecimento.

O algoritmo da ED tem se apresentado robusto e eficiente ao ser testado com sucesso em vários campos da ciência. Por ser uma técnica relativamente nova, ED surgiu em meados da década de 90, muitos pesquisadores têm proposto modificações no algoritmo original da ED com intuito de melhorar sua convergência. Por meio de testes com problemas clássicos de otimização percebeu-se que, às vezes, os resultados obtidos com a ED não são tão bons quanto o esperado. Além disso, em muitos casos, o algoritmo encerra a busca pela solução ótima prematuramente.

Embora os AEs sejam algoritmos que possam resolver uma grande variedade de problemas de otimização, para solucionar problemas complexos, o algoritmo pode demandar um elevado esforço computacional, exigindo muito tempo de processamento. Recentemente, com o avanço e disponibilidade das tecnologias computacionais, tem-se pensando na implementação de algoritmos de otimização em paralelo com o intuito de diminuir este tempo de processamento.

Neste livro, os autores não apenas exploram os fundamentos teóricos da Evolução Diferencial, mas também elevam essa abordagem a novos patamares, apresentando um aprimoramento do método de otimização evolutivo Evolução Diferencial, propondo modificações no algoritmo bá-

sico através da utilização de conjuntos embaralhados (*shuffled complex*) tornando-o capaz de trabalhar com processamento paralelo.

Ao longo destas páginas, os leitores encontrarão uma abordagem abrangente, desde os princípios básicos da Evolução Diferencial até a implementação em ambientes de processamento paralelo. Os casos de estudo e experimentos detalhados servirão como guias valiosos para pesquisadores, profissionais e estudantes que buscam não apenas compreender a teoria, mas também aplicar esses conceitos em suas próprias pesquisas e projetos.

Este livro não é apenas uma contribuição para a literatura científica, mas também uma ponte entre a teoria e a prática, oferecendo insights para aqueles que desejam explorar o potencial da Evolução Diferencial em cenários de processamento paralelo.

Esperamos que este livro inspire novas pesquisas, inovações e aplicações práticas, impulsionando o campo da otimização evolutiva para novos horizontes. Que esta obra sirva como um recurso valioso para todos aqueles que estão comprometidos em avançar o conhecimento e a aplicação prática da Evolução Diferencial no contexto do processamento paralelo.

Ituiutaba, 16 de fevereiro de 2024.

Milena Almeida Leite Brandão
José Laércio Dorício
Sezimária de Fátima Pereira Saramago

Capítulo 1

Problemas de otimização

A otimização matemática, como ramo interdisciplinar da matemática e ciência computacional, concentra-se no desenvolvimento de estudos e ferramentas analíticas e numéricas destinadas a modelar e solucionar problemas complexos de otimização. A sua aplicação é vasta, abrangendo desde questões matemáticas puras até desafios nas ciências físicas, químicas e biológicas, engenharia, arquitetura, economia e gestão, onde se busca encontrar mínimos, máximos ou zeros em funções específicas.

No entanto, a otimização não se limita ao âmbito das ciências; ela é uma realidade presente no cotidiano, tocando quase todos os aspectos da vida. Até mesmo os animais, de maneira instintiva, aplicam princípios de otimização em suas atividades diárias. Um exemplo clássico dessa adaptação é observado nas abelhas. Ao construir os alvéolos das colmeias, utilizados como depósitos para o mel que produzem, esses insetos seguem uma abordagem econômica na forma como moldam essas estruturas. Os alvéolos, que compõem o favo de mel das abelhas europeias, assumem uma forma poliédrica. Nesse processo construtivo, as abelhas procuram a geometria mais eficiente, buscando alcançar o maior volume com a menor quantidade de material utilizado. Em vez de adotarem formatos como cilindros, que resultariam em espaços inaproveitáveis entre os alvéolos, as abelhas optam por uma solução onde os alvéolos assumem a forma de prismas, possibilitando que as paredes de um alvéolo sirvam também aos alvéolos vizinhos. Esse comportamento evidencia a capacidade da natureza em aplicar otimização de maneira intuitiva, um dos muitos exemplos que podemos encontrar na observação do mundo natural.

A otimização é uma ferramenta importante para a tomada de decisão durante a análise e o projeto de sistemas físicos. Para utilizar esta ferramenta torna-se necessário identificar qual o objetivo, ou seja, a quantidade que irá medir a performance do sistema (ex: energia, custo, potência, tempo, temperatura, etc.). O objetivo depende das carac-

terísticas do sistema, que são as *variáveis*, denominadas variáveis de projeto ou variáveis de decisão. Assim, a otimização visa encontrar o valor das variáveis para otimizar o objetivo. Normalmente, estas variáveis devem obedecer a restrições, que são limites impostos ao sistema estudado.

O processo de identificação dos objetivos, variáveis e restrições é denominado *modelagem* do problema. Escrever um modelo adequado é muitas vezes o passo mais importante do processo de otimização. Se o modelo é muito simples, ele pode não representar o problema real. Por outro lado, se o modelo é muito complexo, ele pode trazer muitas dificuldades para a obtenção da solução.

Após a formulação do modelo, um algoritmo de otimização é utilizado para obter a solução do problema, usualmente com a ajuda de códigos computacionais. Não existe um algoritmo universal, mas um conjunto de métodos, entre os quais alguns são mais apropriados para determinadas aplicações específicas. A escolha do método de otimização depende do usuário e pode determinar a eficácia ou falha para a solução do problema.

Após a aplicação do algoritmo o projetista deve ser capaz de analisar a solução encontrada procurando identificar se o processo de otimização obteve sucesso ou se falhou. Em uma análise teórica, existem expressões matemáticas elegantes conhecidas como condições necessárias para existência do ponto ótimo (*optimality condition*) para verificar se a solução atual representa a solução ótima local do problema, mas nos problemas reais elas são difíceis de serem verificadas.

O problema de otimização pode ser escrito como:

$$\begin{array}{ll} \min f(X) & \text{sujeito a } g_k(X) \leq 0 \text{ para } k = 1, 2, \cdots, r \\ X \in \mathbb{R}^n & h_l(X) = 0 \text{ para } l = 1, 2, \cdots, s \end{array} \tag{1.0.1}$$

sendo que,
- X é o vetor das variáveis de decisão;
- f é a função objetivo (ou índice de performance, ou função custo);
- g_k e h_l são funções de restrição (funções escalares de X que definem certas inequações ou equações, respectivamente, que as variáveis devem obedecer).

Existem diversos métodos de otimização e desta forma vários autores já apresentaram diferentes propostas para classificá-los, não sendo possível escolher uma proposta universalmente aceita. Os métodos dependem do tipo do problema em estudo (programação linear ou não-linear, problemas restritos ou irrestritos), das variáveis de projeto (contínuas ou discretas), do tipo de resposta desejada (otimização local ou global; projeto ótimo ou controle ótimo), das técnicas empregadas (determinísticas, estocásticas ou híbridas). Além disso, os problemas podem ser

classificados segundo o número de variáveis (pequenos ou grandes) ou segundo a suavidade das funções (diferenciáveis ou não-diferenciáveis). A seguir estão algumas formas comuns de classificar esses métodos e sugestões de referências bibliográficas:

1. Baseado na natureza do problema:

   - Otimização Contínua: lida com problemas onde as variáveis de decisão x_i assumem valores no conjunto dos números reais ([24], [35], [57]).
   - Otimização Discreta: aplicável a problemas em que as variáveis de decisão devem ser selecionadas a partir de um conjunto discreto de opções. Em alguns problemas de otimização as variáveis de projeto só têm sentido se forem considerados seus valores inteiros (ex: a variável x_i representa o número de caminhões em uma frota de transporte de cargas). Neste tipo de problema, deve-se incluir as "restrições de integralidade", ou seja $x_i \in \mathbb{Z}$ ($\mathbb{Z}$ é o conjunto dos números inteiros). Pode-se também incluir "restrições binárias", neste caso, $x_i \in \{0, 1\}$. Este tipo de problema é denominado problema de otimização inteira ou problemas de otimização discreta. Neste caso, os problemas podem considerar além das variáveis inteiras ou binárias, variáveis abstratas, as quais são representadas por conjuntos finitos ([42], [41], [34]).

2. Baseado nas características da função objetivo e das restrições:

   - Problemas de programação linear: quando a função objetivo e todas as restrições são funções lineares de x. Este tipo de problema é bastante comum em aplicações econômicas e nas áreas de transporte ([55], [7]).
   - Os problemas de programação não-linear são aqueles onde ao menos uma das restrições ou a função objetivo é não-linear; eles aparecem naturalmente nas ciências físicas e nas engenharias ([35], [8], [57]).

3. Baseado na presença de restrições:

   - Otimização irrestrita: problemas de otimização irrestritos são aqueles que não existem funções de restrição. Este caso pode acontecer quando: os problemas práticos não dependem de restrições, as restrições foram desprezadas porque não afetam diretamente a solução ou o problema com restrição foi reescrito de forma que as restrições foram substituídas por funções de penalidade ([35], [57], [10]).

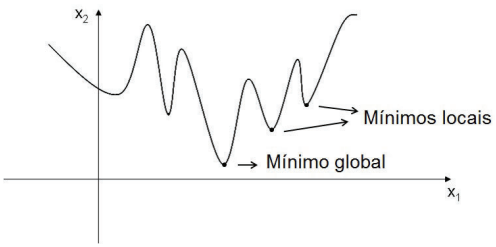
- Os problemas de otimização restritos, que aparecem devido às imposições do projeto, são aqueles onde as variáveis de decisão estão sujeitas às equações ou inequações que devem ser obedecidas. As restrições podem ser lineares (ex: $\sum x_i \leq 1$), laterais (ex: $0 \leq x_i \leq 20$), não-lineares (ex: $\sum x_i^2 - sen x_i \leq 0,5$), ou, ainda, combinações destas ([35], [8], [57], [4]).

Na próxima seção, abordaremos de forma sucinta o Método da Penalidade aplicado a problemas de otimização com restrições. No Capítulo 4, exploraremos a transformação de problemas restritos em problemas irrestritos, utilizando o Método da Penalidade, com o intuito de resolvê-los através algoritmo Evolução Diferencial Melhorada.

4. Baseado na abrangência da busca por soluções ótimas em um espaço de busca:

   - *Otimização global ou local:* considerando um problema de minimização, a solução *global* é aquele ponto onde a função objetivo atinge o *menor* valor entre todos os pontos da região viável. A solução ou *ótimo local* é o ponto cuja função objetivo é menor que todos os pontos vizinhos da região viável, conforme representado na Fig. 1.1. Em muitas aplicações, a solução global é desejada, mas difícil de ser reconhecida. Para problemas *convexos* e na *programação linear* pode-se garantir que uma solução local é também solução global ([23], [27], [50]).

Figura 1.1: Representação de mínimos locais e global



5. Baseado na natureza das variáveis de decisão e na dinâmica do sistema:

   - *Projeto ótimo ou controle ótimo:* segundo [5] o projeto ótimo e o controle ótimo de sistemas são duas atividades distintas.

Porém, tanto o projeto ótimo quanto o controle ótimo são abordagens para alcançar a solução ótima, mas diferem em sua aplicação e foco ([3], [53], [33]).

O projeto ótimo, também conhecido como otimização de projeto, refere-se à busca da melhor configuração ou projeto de um sistema, considerando-se um conjunto de variáveis de projeto. Esse processo visa determinar os valores ótimos dessas variáveis para maximizar ou minimizar uma função objetivo, ao mesmo tempo em que atende a todas as restrições impostas. Em outras palavras, o projeto ótimo visa encontrar a melhor solução em termos de configuração estática ou estrutura do sistema.

Por exemplo, imagine que você queira projetar uma aeronave. O projeto ótimo envolveria determinar a forma, o tamanho, os materiais e outros parâmetros que otimizariam o desempenho da aeronave, como sua velocidade, eficiência de combustível e capacidade de carga, enquanto atendem às restrições de peso, segurança e regulamentações.

Por outro lado, o controle ótimo, também conhecido como otimização de controle, concentra-se em encontrar a melhor trajetória ou estratégia de controle para um sistema dinâmico, ao longo do tempo, com o objetivo de otimizar o desempenho do sistema. Nesse caso, o sistema é governado por leis dinâmicas que descrevem como as variáveis do sistema evoluem com o tempo, e o objetivo é encontrar a melhor sequência de ações de controle para otimizar uma função objetivo.

Por exemplo, considerando novamente a aeronave, o controle ótimo envolveria determinar a melhor sequência de comandos para os atuadores da aeronave (como motores e superfícies de controle) ao longo do tempo, de modo a otimizar a rota, economizar combustível ou minimizar o tempo de voo, enquanto atendem a restrições operacionais, como evitar colisões ou manter a estabilidade da aeronave.

Em resumo, enquanto o projeto ótimo se concentra em encontrar a melhor configuração estática de um sistema, o controle ótimo busca a melhor sequência de ações ao longo do tempo para otimizar o desempenho de um sistema dinâmico. Ambas as abordagens são importantes na otimização e são aplicadas em diferentes contextos, dependendo do problema em questão.

6. Baseado na natureza do algoritmo:

- *Otimização determinística:* os modelos determinísticos de otimização são utilizados quando os parâmetros relevantes para o problema em questão são conhecidos de forma determinística, ou seja, são considerados valores fixos e conhecidos com precisão. Nesse contexto, não há incerteza associada aos valores dos parâmetros. Esses modelos são muito comuns em situações em que os parâmetros do problema podem ser estimados com precisão ou quando os valores são determinados a partir de dados confiáveis ([35], [26], [7], [8]).
- Otimização estocástica: os modelos de otimização estocástica são abordagens matemáticas que lidam com problemas de otimização em que a incerteza está presente nos parâmetros ou nas variáveis do problema. Diferentemente dos modelos determinísticos, em que os parâmetros são conhecidos de forma determinística, os modelos estocásticos reconhecem a natureza aleatória ou incerta de alguns elementos do problema. Esses modelos permitem que os usuários considerem diferentes cenários possíveis e tomem decisões robustas que levem em conta a variabilidade dos dados e a incerteza sobre o futuro ([9], [30], [56]).

  Essas incertezas podem ser representadas por meio de distribuições de probabilidade ou por conjuntos de possíveis valores. Além disso, é necessário definir uma função objetivo e restrições que levem em conta a natureza estocástica do problema.

  Existem várias abordagens para resolver modelos de otimização estocástica. Alguns dos métodos mais comuns incluem: programação estocástica, programação por cenários, programação estocástica robusta, otimização estocástica baseada em simulação e a programação estocástica multiobjetivo.
- *Otimização híbrida:* consiste na utilização conjunta de métodos determinísticos e heurísticos. Tem sido largamente utilizada nos dias atuais, visa unir as boas características de convergência dos métodos determinísticos às vantagens dos métodos heurísticos, por exemplo, a independência do tipo de função e o fato de não ficarem "presos" pelos mínimos locais.

7. Baseado no Número de Objetivos:

   - Otimização Mono-objetivo: refere-se a um tipo de problema de otimização no qual o objetivo é maximizar ou minimizar uma única função objetivo, sujeita a restrições específicas.

Nesse contexto, o termo "mono-objetivo"destaca a presença de apenas uma meta a ser alcançada ([35], [26], [10], [4]).

- Otimização Multiobjetivo: refere-se a um tipo de problema de otimização no qual existem várias funções objetivas a serem consideradas simultaneamente. Diferentemente da otimização mono-objetivo, aqui o desafio é buscar soluções que representem um equilíbrio ou compromisso entre múltiplos objetivos, muitas vezes conflitantes. Essa abordagem reconhece a natureza complexa e multifacetada de muitos problemas do mundo real ([17], [14], [18], [2]).

Essas são apenas algumas das muitas maneiras de classificar métodos de otimização. A escolha da classificação dependerá do contexto específico do problema em questão e dos critérios de análise mais relevantes para o pesquisador ou profissional envolvido em um projeto de otimização.

1.1 Otimização Restrita - Método da Penalidade

Muitos algoritmos de otimização foram desenvolvidos para resolver problemas irrestritos, considerando apenas os limites laterais das variáveis, enquanto apenas alguns algoritmos processam as restrições de desigualdade e igualdade do problema para limitar a região viável. Assim, torna-se necessário utilizar algum artifício para que os problemas com restrições também se apliquem aos métodos de otimização irrestrita.

Uma das abordagens fundamentais para a otimização restrita é substituir o problema original por uma função de penalidade que é composta da função objetivo original do problema de otimização com restrições somada a um termo adicional para cada restrição, o qual é positivo quando o ponto atual X viola essa restrição e zero caso contrário.

A maioria das abordagens definem uma sequência de funções de penalidade, na qual os termos penalizados por violarem as restrições são multiplicados por um coeficiente positivo. Ao fazer este coeficiente crescer, penalizam-se as violações das restrições mais severamente, forçando o ponto mínimo da função de penalidade se aproximar cada vez mais da região viável do problema restrito.

Essas abordagens são conhecidas como Método da Penalidade Exterior, porque o termo penalizado por cada restrição é diferente de zero somente quando X é inviável em relação a tal restrição. Segundo [57], muitas vezes, o ponto de mínimo das funções de penalidade são inviáveis com relação ao problema original, e aproximam-se da viabilidade apenas no limite em que os parâmetros de penalidade se tornam cada vez maiores.

Considere o seguinte problema:

$$\begin{array}{lll} \min f(X) & \text{sujeito a} & g_j(X) \leq 0, \quad \text{restrições de desigualdade} \\ X \in \mathbb{R}^n & & h_l(X) = 0, \quad \text{restrições de igualdade} \end{array} \tag{1.1.2}$$

A fim de que os problemas restritos (1.1.2) sejam transformados em problemas irrestritos será utilizado, neste estudo, o Método da Penalidade Exterior. Se o objetivo da otimização é minimizar a função objetivo $f(X)$, a função de penalidade $P(X)$ é somada a função principal $f(X)$ de modo a aumentar o valor da função nos pontos que estão fora da região viável, por outro lado, se o objetivo é maximizar, a função de penalidade é subtraída da função principal se o ponto não obedece às restrições.

Esta nova função objetivo, chamada *pseudo objetivo*, é penalizada de acordo com um *fator de penalidade* r_p toda vez que encontrar uma restrição ativa. Assim, este escalar amplia a penalidade, e quanto maior for seu valor, maior será a eficiência do método para obedecer às restrições. O modelo matemático da função pseudo objetivo $\Phi(X)$ utilizada para minimizar uma função $f(X)$ é dada por:

$$\Phi(X) = f(X) + r_p P(X), \tag{1.1.3}$$

e a função de penalidade $P(X)$ dada por:

$$P(X) = \sum_{j=1}^{m} \left(max\left[0, g_j(X)\right]^2 \right) + \sum_{k=1}^{l} h_l(X)^2, \tag{1.1.4}$$

sendo $g_j(X)$ as restrições de desigualdade, $h_l(X)$ as restrições de igualdade e m e l o número de restrições de desigualdade e igualdade respectivamente.

Deste modo, os pontos fora do espaço viável apresentarão valores muito ruins para a otimização e serão facilmente desconsiderados pelos algoritmos de otimização irrestrita. Para uma consideração mais detalhada do Método da Penalidade veja [35].

Capítulo 2

Evolução Diferencial Melhorada

O algoritmo da Evolução Diferencial Melhorada (EDM) proposto é uma combinação da evolução com conjuntos embaralhados e a Evolução Diferencial clássica (ED). Os métodos de evolução com conjuntos embaralhados e evolução diferencial estão descritos com detalhes em [12]. Porém, visto que será preciso o uso de algumas equações e definições para construir o método EDM, em seguida será apresentado um resumo do método ED.

2.1 Evolução Diferencial

O algoritmo da ED começa criando uma população inicial de N_p indivíduos, escolhida aleatoriamente e devendo cobrir todo o espaço de busca. Geralmente, é criada por uma distribuição de probabilidade uniforme, quando não há nenhum conhecimento sobre o problema.

Cada indivíduo, chamado de *vetor*, possui n componentes representadas por valores reais, sendo n o número de variáveis de projeto. Assim, a população segue uma evolução natural, em que o número de indivíduos é constante durante todas as gerações.

A ideia principal da evolução diferencial é gerar novos indivíduos, denotados vetores modificados ou doadores, pela adição da diferença ponderada entre dois indivíduos aleatórios da população a um terceiro indivíduo. Esta operação é chamada *mutação*. As componentes do indivíduo doador são misturadas com as componentes de um indivíduo escolhido aleatoriamente (denotado vetor alvo), para resultar o chamado vetor tentativa, ou vetor experimental. O processo de misturar os parâmetros é referido como *cruzamento*. Se o vetor experimental resultar um valor da função objetivo menor que o vetor alvo, então o vetor experimental substitui o vetor alvo na geração seguinte. Esta úl-

tima operação é chamada *seleção*. O procedimento é finalizado quando algum critério de parada é atingido.

Considere o seguinte problema de otimização sem restrições (a menos das laterais):

$$\min \quad f(X), \quad X \in \mathbb{R}^n \tag{2.1.1}$$

onde $f(X)$ representa a função objetivo e sejam $X^L = (x_1^L, x_2^L, \cdots, x_n^L)^T$ e $X^U = (x_1^U, x_2^U, \cdots, x_n^U)^T$ as restrição laterais inferior e superior, respectivamente.

A população de indivíduos durante a q-ésima geração é definida como:

$$X_d^{(q)} = (x_{d,1}, x_{d,2}, \cdots, x_{d,n})^T, \qquad d = 1, \cdots, N_p \tag{2.1.2}$$

A população inicial $X_d^{(0)}$ é escolhida aleatoriamente, dentro do espaço de busca, definido pelas restrições laterais, fornecidas pelo usuário:

$$x_{d,i}^{(0)} = x_i^L + r(x_i^U - x_i^L), \qquad i = 1, \cdots, n \quad e \quad d = 1, \cdots, N_p \tag{2.1.3}$$

sendo que r é um número gerado randomicamente entre 0 e 1.

Mutação

Sejam os vetores X_α, X_β e X_γ escolhidos aleatoriamente e distintos entre si. Na geração q um par de vetores (X_β, X_γ) define uma diferença $X_\beta - X_\gamma$. Esta diferença é multiplicada por $F > 0$, sendo denotada por diferença ponderada, e é usada para perturbar o terceiro vetor X_α ou o melhor vetor X_{best} da população. Este processo, que resulta no vetor doador $V^{(q+1)}$, pode ser escrito matematicamente como:

$$\begin{aligned} &V^{(q+1)} = X_\alpha^{(q)} + F\left(X_\beta^{(q)} - X_\gamma^{(q)}\right) \\ &ou \\ &V^{(q+1)} = X_{best}^{(q)} + F\left(X_\beta^{(q)} - X_\gamma^{(q)}\right) \end{aligned} \tag{2.1.4}$$

sendo que os índices aleatórios $\alpha, \beta, \gamma \in \{1, \cdots, N_p\}$ são inteiros distintos entre si e diferentes do índice d. O número de indivíduos da população, N_p, deve ser maior ou igual a 4. O fator de escala, ou taxa de perturbação, F, é um fator real, positivo e constante variando entre 0 e 2, cujo objetivo é controlar a amplitude da diferença ponderada.

Se o número de indivíduos da população é grande o suficiente, a diversidade da população pode ser melhorada usando duas diferenças ponderadas para perturbar um vetor existente. Este processo pode ser dado por:

$$\begin{aligned} &V^{(q+1)} = X_\alpha^{(q)} + F\left(X_\lambda^{(q)} - X_\beta^{(q)} + X_\gamma^{(q)} - X_\delta^{(q)}\right) \\ &ou \\ &V^{(q+1)} = X_{best}^{(q)} + F\left(X_\lambda^{(q)} - X_\beta^{(q)} + X_\gamma^{(q)} - X_\delta^{(q)}\right) \end{aligned} \tag{2.1.5}$$

onde os índices aleatórios $\alpha, \beta, \delta, \gamma, \lambda \in \{1, \cdots, N_p\}$, são inteiros mutuamente distintos e diferentes do índice d, tais que $Np \geq 6$.

Cruzamento

Na operação cruzamento, cujo objetivo é aumentar a diversidade dos indivíduos que sofreram a mutação, as componentes do vetor doador, $V^{(q+1)}$, são misturadas com as componentes de outro indivíduo denominado vetor alvo. A escolha do vetor alvo, que deve ser diferente dos vetores já usados anteriormente, é aleatório, segundo uma probabilidade de cruzamento CR. Desta forma, obtém-se o vetor tentativa ou experimental, $U^{(q+1)}$, definido como:

$$u_i^{(q+1)} = \begin{cases} v_i^{(q+1)}, & \text{se } r_i \leq CR \\ \\ x_{d,i}^{(q)}, & \text{se } r_i > CR, \quad i = 1, \cdots, n. \end{cases} \tag{2.1.6}$$

onde r_i é um número randômico entre 0 e 1 e $x_{d,i}$ são as componentes do vetor alvo $X_d^{(q)}$. A probabilidade do cruzamento ocorrer, CR, representa a probabilidade do vetor experimental herdar os valores das variáveis do vetor doador, e está compreendida entre 0 e 1, sendo fornecida pelo usuário.

O cruzamento dado pela Eq. (2.1.6) é denominado de *cruzamento binomial*.

Outra forma de realizar o cruzamento, denominado *cruzamento exponencial*, no qual as componentes do vetor experimental são dadas pelas componentes do vetor doador enquanto o número randômico for menor ou igual à probabilidade de cruzamento CR.

Seleção

A seleção é o processo de produzir melhores filhos. O custo do vetor experimental $U^{(q+1)}$ é calculado e comparado com o custo do vetor alvo $X_d^{(q)}$. Este processo pode ser escrito como:

$$\begin{cases} \text{Se } f(U^{(q+1)}) \leq f(X_d^{(q)}), \text{ então } X_d^{(q+1)} = U^{(q+1)} \\ \\ \text{Se } f(U^{(q+1)}) > f(X_d^{(q)}), \text{ então } X_d^{(q+1)} = X_d^{(q)} \end{cases} \tag{2.1.7}$$

O processo iterativo continua até que seja alcançado algum critério de parada.

2.2 Evolução Diferencial Melhorada

A EDM inicia-se como o algoritmo da ED usual por criar uma população de indivíduos amostrados aleatoriamente a partir da região viável

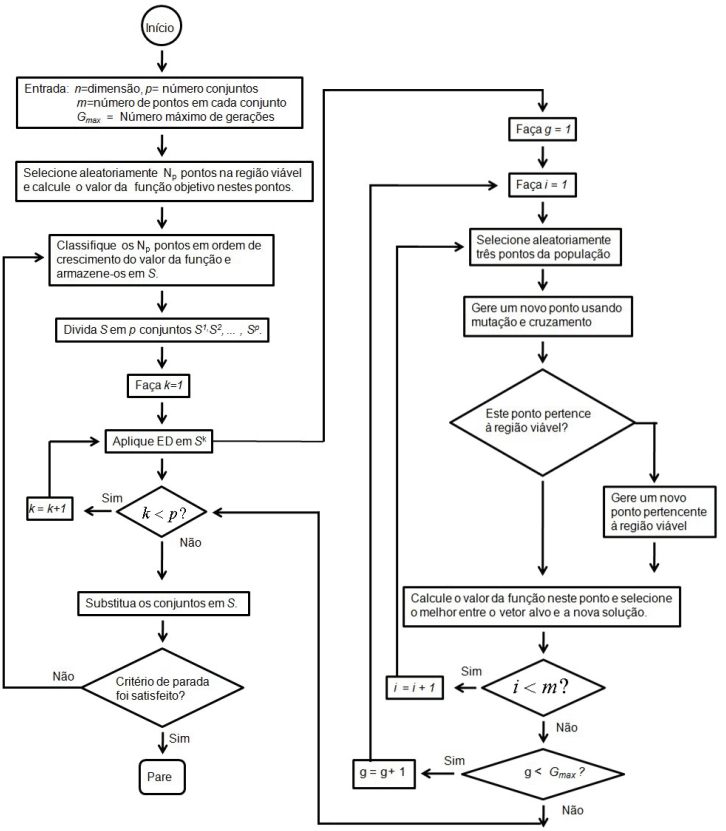


Figura 2.1: Fluxograma do Método Evolução Diferencial Melhorada - EDM.

usando distribuição de probabilidade uniforme. A população é então classificada em ordem crescente de valores da função objetivo e particionada em diversos conjuntos. Cada conjunto independentemente executa a ED. Na etapa da evolução, os conjuntos são obrigados a se misturar e os pontos são realocados para garantir a troca de informações. O processo do algoritmo EDM proposto é descrito a seguir e os seu fluxograma é apresentado na Fig. 2.1.

Passo 1: *Inicialização.* Gerar aleatoriamente uma população inicial de N_p vetores $X_{i,j}$, sendo que cada vetor tem dimensão n, usando a seguinte regra:

$$X_{i,j} = X_{min,j} + rand(X_{max,j} - X_{min,j}), \tag{2.2.8}$$

onde $X_{min,j}$ e $X_{max,j}$ são os limites inferior e superior para a j-ésima

componente respectivamente e *rand* um número aleatório uniforme entre 0 e 1. Calcule o valor da função objetivo $f_i = f(X_i)$ para todo X_i. Defina o número máximo de geração como G_{max}. Seja $N_p = p \times m$, onde p é o número de conjuntos e m é o número de indivíduos em cada conjunto.

Passo 2: *Classificação.* Classifique a população inteira em ordem de crescimento do valor da função objetivo. Armazene-os em um conjunto $S = \{X_i, f_i : i = 1, \cdots, N_p\}$.

Passo 3: *Divisão.* Divida S em p subpopulações $S^1, S^2, \cdots, S^p$, cada uma contendo m pontos, tais que: $S^k = \{X_j^k, f_j^k : X_j^k = X_{k+p(j-1)}, f_j^k = f_{k+p(j-1)}, j = 1, \cdots, m\}, k = 1, 2, \cdots, p$.

Passo 4: *Evolução.* Seja $k = 1$.

Passo 4.1: Aplique ED a cada subpopulação S^k.

Passo 4.2: *Inicialização.* Defina o contador da geração $g = 1$. O conjunto S^k trabalha como uma população em ED.

Passo 4.3: *Mutação.* Selecione aleatoriamente três pontos da população S^k e gere o vetor doador V_i utilizando a Eq. (2.1.4).

Passo 4.4: *Cruzamento.* Recombine cada vetor alvo X_i com o vetor doador gerado no passo 4.3 para gerar o vetor experimental U_i utilizando a equação (2.1.6).

Passo 4.5: *Verificação de viabilidade.* Verifique se cada variável do vetor experimental pertence à região viável. No caso afirmativo, vá para o Passo 4.6, caso contrário, corrija-o da seguinte forma:

$$U_{i,j} = \begin{cases} 2X_{min,j} - U_{i,j} & \text{se } U_{i,j} < X_{min,j} \\ \\ 2X_{max,j} - U_{i,j} & \text{se } U_{i,j} > X_{max,j} \end{cases} \tag{2.2.9}$$

e vá para o Passo 4.6.

Passo 4.6: *Seleção.* Calcule o valor da função objetivo para o vetor U_i. Escolha o melhor vetor comparando o valor da função objetivo dos vetores alvo e experimental usando a Eq. (2.1.7) para a próxima geração.

Passo 4.7: *Iteração.* Se $g < G_{max}$ então vá para o passo 4.3 com $g = g + 1$, caso contrário, vá para o passo 5.

Passo 5: Se $k < p$ então $k = k + 1$ e vá para o passo 4.1, caso contrário, vá para o Passo 6.

Passo 6: *Embaralhando os conjuntos.* Substitua os conjuntos $S^1, S^2, \cdots, S^p$ em S e verifique se os critérios de parada foram satisfeitos, se sim, pare, caso contrário vá para o Passo 2.

A partir do algoritmo apresentado várias outras versões já foram propostas alterando o número de conjuntos (subpopulações) utilizados e também o número de vezes que o algoritmo da ED básica é chamado para cada conjunto antes do processo de embaralhamento. A arquitetura do método EDM é ilustrado na Fig. 2.2.

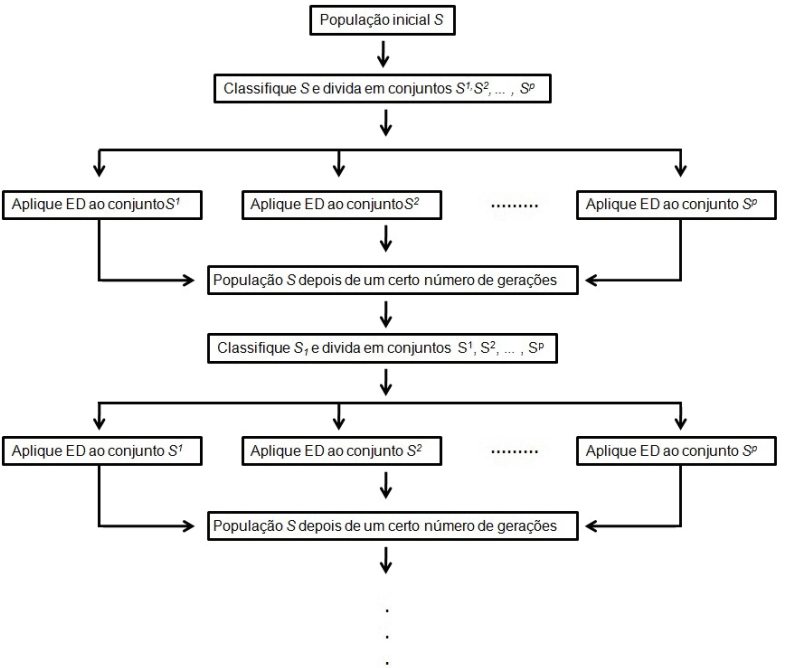


Figura 2.2: Arquitetura do Método Evolução Diferencial Melhorada - EDM.

2.3 O algoritmo Evolução Diferencial Melhorada implementado em processamento paralelo

Em uma sociedade capitalista envolta de curtos prazos e visando sempre o lucro, quanto mais rápido o algoritmo conseguir resolver um problema melhor. A ideia de se dividir tarefas de programas por vários processadores é antiga, mas só recentemente vem se tornando viável devido ao rápido avanço de *hardware* e *software*. As máquinas evoluíram muito em velocidade e com o aparecimento de máquinas com vários processadores a velocidade de comunicação vem apresentando uma grande evolução permitindo que estes processadores troquem informações com rapidez; e os programas que possibilitam esta comunicação vêm mostrando grande aumento de eficiência. Dessa forma, o objetivo é diminuir o tempo de processamento durante a execução do algoritmo melhorado, para permitir sua aplicação em problemas complexos.

Agora é necessário combinar as potencialidades dos métodos da Evolução Diferencial e da Evolução com Conjuntos Embaralhados (ECE) para implementar o algoritmo da EDM em paralelo, pois assim, cada conjunto do método ED evoluirá em um processador diferente e ao mesmo tempo, seguindo o princípio de que grandes problemas geralmente podem ser divididos em problemas menores, para então serem resolvidos concorrentemente (em paralelo).

No processador mestre é criado randomicamente a população inicial, a qual é divida em k subpopulações ($k \in \mathbb{Z}$ é menor ou igual ao número de processadores) e distribuída pelos no máximo k processadores. Em seguida, o código da ED segue processando sequencialmente em cada processador até atingir algum critério de parada. As subpopulações são então reagrupadas no processador mestre, embaralhadas e divididas novamente em subpopulações.

Resumidamente, o método de otimização Evolução Diferencial Melhorada implementado em processamento paralelo (EDMP) une o que há de melhor nos métodos ED e ECE em um só algoritmo que é implementado em paralelo. O fluxograma do Método de Otimização Evolução Diferencial Melhorada em Paralelo pode ser visto na Fig. 2.3.

O algoritmo da EDMP foi implementado em C++ e suas primeiras simulações foram realizadas em um Cluster Beowulf de 36 processadores.

O primeiro e mais importante passo para escrever o código da EDM em paralelo foi decidir em quais momentos haveria comunicação entre os processadores pois quanto maior for a comunicação menor é a eficiência do algoritmo em paralelo. Novamente, foi utilizado a ideia de dividir a população inicial em complexos ou subpopulações. Em seguida, cada subpopulação iria evoluir separadamente usando a ED até

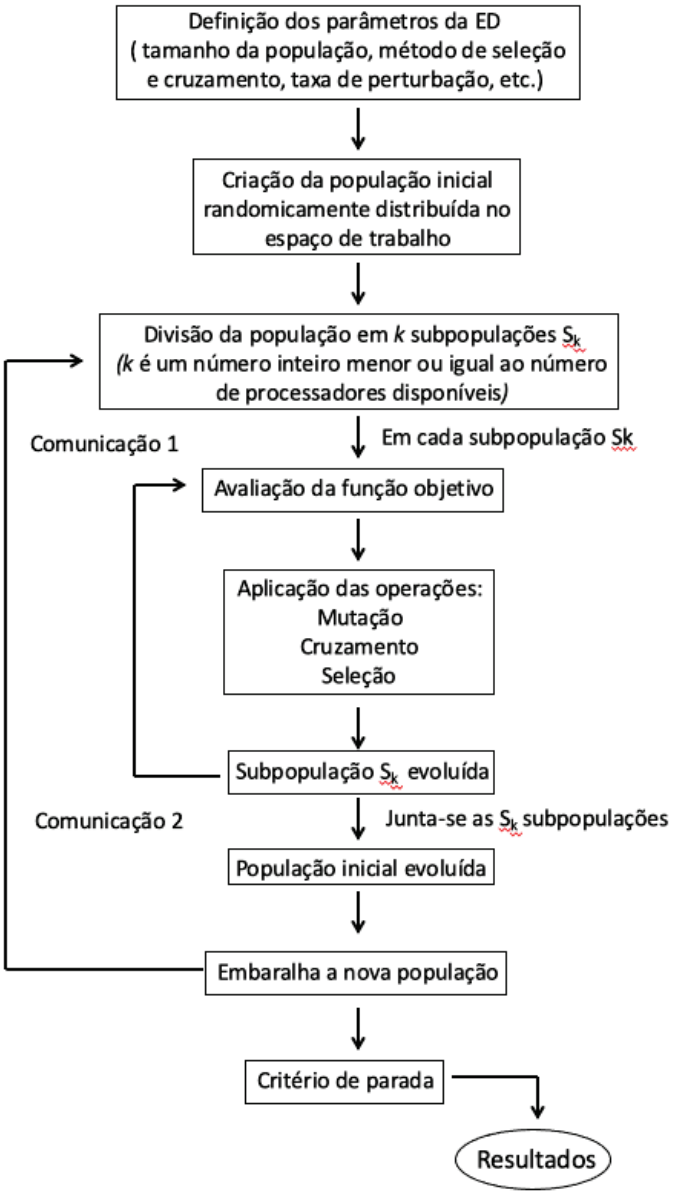


Figura 2.3: Algoritmo Evolução Diferencial Melhorada implementado em processamento paralelo.

que algum critério de parada referente a ED fosse satisfeito. A seguir, as subpopulações seriam reagrupadas, embaralhadas e dividas novamente em subpopulações e o processo continuaria até satisfazer algum critério de parada da EDM.

No processador mestre é criado randomicamente a população inicial, a qual é dividida em k subpopulações ($k \in \mathbb{Z}$ é menor ou igual ao número de processadores) e distribuída pelos no máximo k processadores, a primeira comunicação entre os processadores acontece aqui. Em seguida, o código da ED segue processando sequencialmente em cada processador até atingir algum critério de parada. As subpopulações são então reagrupadas no processador mestre, que é a segunda comunicação entre os processadores, embaralhada e dividida novamente em subpopulações. Essas duas comunicações entre os processadores são feitas por meio do comando MPI_allgatherv, cuja função é reunir os dados e distribuir os dados combinados de todas as tarefas.

Algumas verificações foram realizadas utilizando funções testes a fim de se encontrar possíveis erros de implementação bem como verificar a capacidade de otimização do código. As funções testes de otimização utilizadas possuem características distintas: restritas, irrestritas, multimodais e uni-modais. Todos os problemas testes usados para validação do código foram consultados no site [40] e suas simulações serão apresentadas no Capítulo 4.

Depois de vários testes, percebeu-se que para alguns problemas o algoritmo não conseguia "escapar" dos mínimos locais. Ao listar os membros da população notou-se que esta tornava-se muito homogênea, as subpopulações provenientes de processadores diferentes algumas vezes eram praticamente iguais. Para evitar isso, em cada nova iteração da EDM ao invés de juntar todas as subpopulações em uma só, 50% dos membros da nova população proviam das subpopulações evoluídas e os outros 50% seriam gerados aleatoriamente dentro da região viável do problema. Desta forma, conseguiu-se evitar a parada prematura do algoritmo por aumentar a diversidade da população.

Novamente, foram utilizadas as funções testes a fim de validar o código, que após várias análises não apresentou problemas de implementação nem de estagnação, confirmando assim sua eficácia.

Para facilitar a compreensão será resolvido um problema de otimização ilustrativo. Os dados apresentados neste caso são para a versão sequencial do algoritmo da EDMP, ou seja, todas as etapas serão executadas em um único processador.

Exemplo 2.1. *Seja o problema de minimizar uma função, sem restrição, a menos de laterias, dada por:*

$$f(x, y) = xsen(4x) + 1,1ysen(2y), \qquad 8 < x, y < 10$$

A Fig. 2.3 mostra o gráfico da superfície e as curvas de nível da função f dada no exemplo 2.1. É possível observar que essa função tem um mínimo bem definido e é esse valor que será encontrado via EDMP.

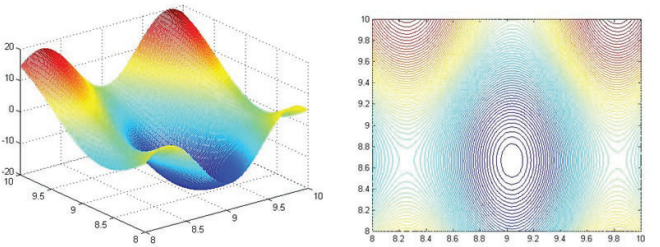


Figura 2.4: Gráfico e curvas de nível da função dada no exemplo 2.1

Como o programa computacional da EDMP faz a minimização, a função objetivo será escrita como: $min \quad f(x,y) = [xsen(4x) + 1,1ysen(2y)]$.

A população inicial será criada apenas com 6 indivíduos para possibilitar a visualização de todas as etapas. A Fig. 2.3 mostra a população inicial criada pelo algoritmo. A primeira coluna são os valores de x_i, a segunda coluna são os valores de y_i e a terceira coluna os valores de $f(x_i, y_i)$, para $i = 0, 1, \cdot, 5$. O *Randperm* é o sorteio aleatório dos elementos que participarão na etapa da mutação. Conforme a figura, os indivíduos sorteados foram (x_0, y_0) e (x_1, y_1). A *Matriz x* separa os indivíduos que sofrerão mutação. E por último, *bm* é o melhor elemento da população, já que o seu valor na função objetivo é o menor e estamos minimizando f.

Todos os indivíduos da população participarão da etapa do cruzamento na ordem que aparecem na lista dos indivíduos e serão chamados de vetores alvo nessa operação.

Assim, a mutação ocorrerá da seguinte forma:

$$V = X_{best} + F(X_\beta - X_\gamma)$$

$$(9.17005, 8.95978) + F\left((9.17005, 8.95978) - (8.0025, 9.12721)\right)$$

O que resultará, adotando $F = 0.8$, no vetor doador $V_0 = (10.10409, 8.825836)$.

Em seguida, realizar-se-á o cruzamento binomial entre o primeiro vetor (indivíduo) da população e o vetor doador obtido na mutação. Pelo sorteio apresentado na Fig. 2.3 as características do vetor experimental (filho) virão todas do vetor doador, ou seja, o vetor experimental será igual ao vetor doador.

```
Matriz s:
  9.17005   8.95978  -15.7092
   8.0025   9.12721  -1.14928
   8.7006   9.79198   5.10872
  9.64573   9.49326   8.88339
  8.38662   9.61753   11.0848
  8.34823   9.71794   13.5836

Randperm: 0 1 4 2 3 5


Matriz x:
  9.17005   8.95978  -15.7092
   8.0025   9.12721  -1.14928


bm   = 9.17005 8.95978 -15.7092
```

Figura 2.5: População inicial do exemplo 2.1.

Visto que a primeira coordenada do vetor experimental assume valor fora dos limites laterais, o valor dessa coordenada será corrigido para o valor do limite lateral mais próximo desta coordenada. Após correção o valor do vetor experimental é $U_1 = (10, 8.82583)$ e $f(U_1) = -1.59006$

Após o cruzamento, será feita a seleção de qual indivíduo permanecerá na população, se o vetor alvo ou o vetor experimental. A seleção é feita escolhendo o indivíduo que possui o menor custo, isto é, cujo valor na função objetivo é o menor. Comparando os valores, percebe-se que o vetor alvo permanecerá na população e o vetor experimental será eliminado. Agora, todo esse processo é repetido até que todos os indivíduos da população participem da operação de cruzamento. O próximo indivíduo que participará do processo de evolução é o segundo indivíduo da lista da Fig. 2.3, o vetor $X_1 = (8.0025, 9.12721)$. As Figs. 2.3 e 2.3 mostram a mutação e seleção, respectivamente, da segunda iteração.

Em seguida, será apresentada a mutação e seleção de cada uma das iterações restantes: terceira, quarta e quinta.

Aqui finaliza-se o processo de evolução da primeira geração. O processo iterativo continua até que seja alcançado algum critério de parada, tais como:

- Número máximo de gerações;
- Avaliação da diferença do valor da função objetivo da geração

```
Cruzamento vindo do vetor alvo:  binomial
0 0

Cruzamento vindo do vetor doador:
1 1

Vetor alvo:
9.17005 8.95978 -15.7092

Vetor doador:
10.1041 8.82583

Vetor experimental antes dos limites laterais:
10.1041 8.82583

Vetor experimental apos limites laterais:
10 8.82583 -1.59006
```

Figura 2.6: Primeiro cruzamento da população, envolvendo o vetor alvo X_0.

atual e da anterior;

- Laço externo, entre outros.

A Fig. 2.3 mostra a população evoluída e o início da evolução da segunda geração.

```
Matriz s:
  9.17005   8.95978  -15.7092
   8.0025   9.12721  -1.14928
   8.7006   9.79198   5.10872
  9.64573   9.49326   8.88339
  8.38662   9.61753   11.0848
  8.34823   9.71794   13.5836

Randperm: 0 1 2 4 3 5


Matriz x:
  9.17005   8.95978  -15.7092
   8.0025   9.12721  -1.14928


bm   = 9.17005 8.95978 -15.7092
```

Figura 2.7: Mutação da segunda iteração.

```
Cruzamento vindo do vetor alvo:  binomial
0 1

Cruzamento vindo do vetor doador:
1 0

Vetor alvo:
8.0025 9.12721 -1.14928

Vetor doador:
10.1041 8.82583

Vetor experimental antes dos limites laterais:
10.1041 9.12721

Vetor experimental apos limites laterais:
10 9.12721 1.82245
```

Figura 2.8: Seleção da segunda iteração.

```
Matriz s:
  9.17005   8.95978  -15.7092
   8.0025   9.12721  -1.14928
  8.79449   9.62554 -0.975158
  9.64573   9.49326   8.88339
  8.38662   9.61753   11.0848
  8.34823   9.71794   13.5836

Randperm: 4 0 1 3 2 5


Matriz x:
  8.38662   9.61753   11.0848
  9.17005   8.95978  -15.7092


bm   = 9.17005 8.95978 -15.7092
```

Figura 2.9: Mutação da terceira iteração.

```
Cruzamento vindo do vetor alvo:  binomial
0 0

Cruzamento vindo do vetor doador:
1 1

Vetor alvo:
9.64573 9.49326 8.88339

Vetor doador:
8.54331 9.48598

Vetor experimental antes dos limites laterais:
8.54331 9.48598

Vetor experimental apos limites laterais:
8.54331 9.48598 4.47694

Vetor alvo atualizado: 4.47694
```

Figura 2.10: Seleção da terceira iteração.

```
Matriz s:
  9.17005   8.95978  -15.7092
   8.0025   9.12721  -1.14928
  8.79449   9.62554 -0.975158
  8.54331   9.48598   4.47694
  8.38662   9.61753   11.0848
  8.34823   9.71794   13.5836

Randperm: 1 3 4 0 2 5


Matriz x:
   8.0025   9.12721  -1.14928
  8.54331   9.48598   4.47694


bm   = 9.17005 8.95978 -15.7092
```

Figura 2.11: Mutação da quarta iteração.

```
Cruzamento vindo do vetor alvo:  binomial
0 1

Cruzamento vindo do vetor doador:
1 0

Vetor alvo:
8.38662 9.61753 11.0848

Vetor doador:
8.73741 8.67276

Vetor experimental antes dos limites laterais:
8.73741 9.61753

Vetor experimental apos limites laterais:
8.73741 9.61753 0.639078

Vetor alvo atualizado: 0.639078
```

Figura 2.12: Seleção da quarta iteração.

```
Matriz s:
  9.17005   8.95978  -15.7092
   8.0025   9.12721  -1.14928
  8.79449   9.62554 -0.975158
  8.54331   9.48598   4.47694
  8.73741   9.61753  0.639078
  8.34823   9.71794   13.5836

Randperm: 2 3 0 1 4 5


Matriz x:
   8.79449    9.62554  -0.975158
   8.54331    9.48598    4.47694


bm   = 9.17005 8.95978 -15.7092
```

Figura 2.13: Mutação da quinta iteração.

```
Cruzamento vindo do vetor alvo:  binomial
0 0

Cruzamento vindo do vetor doador:
1 1

Vetor alvo:
8.34823 9.71794 13.5836

Vetor doador:
9.371 9.07142

Vetor experimental antes dos limites laterais:
9.371 9.07142

Vetor experimental apos limites laterais:
9.371 9.07142 -8.47959

Vetor alvo atualizado: -8.47959
```

Figura 2.14: Seleção da quinta iteração.

```
 ********** g = 2 k = 0 **********

Matriz s:
  9.17005   8.95978  -15.7092
   8.0025   9.12721  -1.14928
  8.79449   9.62554 -0.975158
  8.54331   9.48598   4.47694
  8.73741   9.61753  0.639078
    9.371   9.07142  -8.47959

Randperm: 2 1 3 4 0 5


Matriz x:
   8.79449    9.62554  -0.975158
    8.0025    9.12721   -1.14928


bm   = 9.17005 8.95978 -15.7092
```

Figura 2.15: Início da segunda geração da população.

Continuando o processo, após 610 avaliações da função objetivo, tem-se a solução ótima do problema dado em 2.1. Os valores encontrados são apresentados na Fig. 2.3.

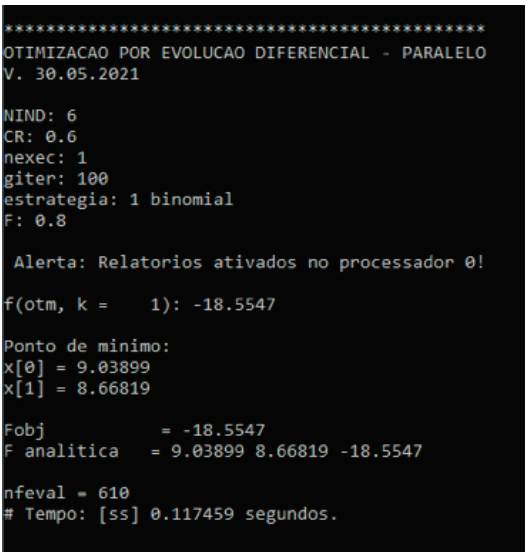


Figura 2.16: Solução ótima do problema dado em 2.1.

A solução do problema dado em 2.1 também pode ser obtida no site do WolframAlpha conforme Fig. 2.3.

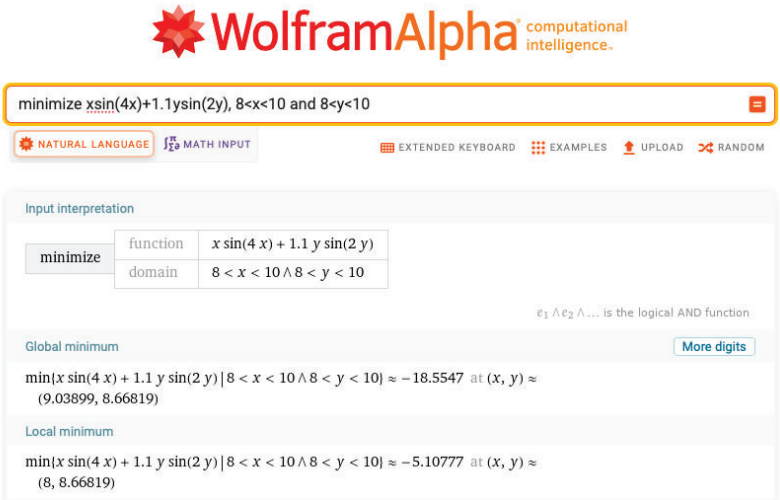


Figura 2.17: Solução ótima do problema dado em 2.1 obtida no WolframAlpha.

Capítulo 3

Noções sobre processamento paralelo

O processador, seja ele em computadores, note/netbooks, tablets ou GPS's, é uma das principais ferramentas para se solucionar problemas no dia a dia, principalmente quando estes problemas estão relacionados à criação ou modelagem de sistemas mecânicos ou físicos encontrados em várias aplicações. A facilidade para se implementar algoritmos matemáticos, utilizando linguagem de programação de alto nível, tem levado a comunidade científica a estudar problemas cada vez mais complexos.

Um cluster é um sistema que atende às aplicações que podem ser divididas em tarefas, que poderão, assim, ser executadas simultaneamente por um certo número de processadores. Segundo a [1], praticamente todos os setores estão adotando clusters Linux a fim de obter as melhorias de desempenho necessárias para cumprir metas organizacionais. Ainda segundo [1], a computação de alto desempenho é aproveitada em áreas como análise sísmica para exploração de petróleo, simulação aerodinâmica para projetos de motores e aeronaves, efeitos especiais de Hollywood, modelagem molecular para pesquisa biomédica, computação empresarial super-escalável e mineração de dados e modelagem financeira para análise de negócios.

Sabe-se que não existe um algoritmo universal que resolva qualquer tipo de problema. Além disso, alguns algoritmos demandam um tempo de processamento elevado e muitos recursos computacionais para encontrar algum resultado. Algumas vezes não se conhece formas para melhorar a performance do algoritmo que está sendo aplicado, outras vezes o algoritmo implementado é um código comercial, o que impossibilita modificações estruturais no algoritmo. Nestes casos, pode-se recorrer a outros recursos para melhorar o desempenho do código. Uma forma de fazer isto é por meio da computação paralela.

A busca por maior eficiência computacional no processamento em computação de alto desempenho (*High Performance Computing* - HPC), medida em termos do tempo necessário a obter-se a solução de um problema, tem estimulado a procura por novas estratégias para se alcançar elevadas capacidades de processamento. Atualmente, há, pelo menos, duas alternativas para a HPC, uma delas é a utilização dos super computadores, que são unidades de processamento de grande desempenho e a outra são vários computadores em paralelos. Porém, o custo de implantação e manutenção de super computadores é altíssima, por este motivo a utilização de computadores pessoais (PCs) para formar unidades de processamento em paralelo tem sido, hoje, a alternativa mais atraente.

Como o objetivo de um algoritmo paralelo é ter um tempo de execução inferior à versão sequencial é útil que se crie primeiro a versão sequencial do algoritmo do problema que se pretende resolver para então adaptá-lo à versão paralela ou, se possível, que se use o algoritmo serial como ponto de partida, pois desta forma, além de ajudar a compreender o problema o algoritmo serial também pode auxiliar na validação do algoritmo em paralelo.

Para se obter algoritmos paralelos eficientes é preciso considerar, pelo menos, cinco fatores importantes, a saber: o balanceamento de carga, isto é, dividir de forma igual o trabalho entre os processadores; minimizar as necessidades de comunicação; minimizar o tempo ocioso; sobrepor as operações de comunicação e de computação e finalmente, mas não menos importante, minimizar as operações de entrada e saída (E/S). Estas questões serão consideradas mais adiante. Também é necessário decidir se o paralelismo será com memória distribuída ou compartilhada. No primeiro caso, a comunicação é feita por meio do acesso à memória principal e por isso limita-se a um número relativamente pequeno de processadores. No paralelismo com memória distribuída cada processador tem acesso somente à memória a ele alocada e o acesso aos dados da memória de outros processadores ocorre via troca de mensagens entre os processadores por meio de uma rede de conexão. Neste trabalho será utilizado paralelismo com memória distribuída.

Visto que há um alto custo em se utilizar multiprocessadores os algoritmos apresentados aqui serão implementados em clusters de computadores pessoais que são economicamente mais viáveis e com poder de processamento próximo ao das máquinas paralelas.

3.1 Cluster

Segundo [16] o termo "cluster de computadores refere-se a utilização de diversos computadores (heterogêneos ou não) conectados em rede

para realização de processamento paralelo. Ou seja, as máquinas são conectadas via rede para formar um único computador". Desta forma, o cluster dá aos usuários a impressão de um único sistema funcionando quando na verdade podem haver dezenas, centenas ou até milhares de computadores. Dessa forma, é possível realizar processamentos que até então somente computadores de alta performance seriam capazes de fazer.

Cada computador de um cluster é denominado nó, há um nó sevidor e os outros são chamados de nós clientes, os quais são interconectados via rede que pode ser Ethernet ou outra topologia qualquer. É interessante que esta rede seja criada de forma a permitir o acréscimo ou a retirada de um nó sem, no entanto, interromper o funcionamento do cluster, pois se houver danos a um ou mais computadores, por exemplo, é possível realizar os reparos sem prejudicar o processamento. Além disso, o sistema operacional usado nos computadores deve ser o mesmo, isto é, ou somente Windows, ou somente Linux, por exemplo, visto que há particularidades em cada sistema operacional que poderiam impedir o funcionamento do cluster.

Depois de escolhido o sistema operacional, é preciso usar um software que permita a implementação de estruturas de Máquinas Virtuais Paralelas (PVM - *Parallel Virtual Machines*) e/ou Interface de Passagem de Mensagens (MPI - *Message Passing Interface*). Esse software será responsável por controlar as trocas de mensagens do cluster, distribuir os arquivos e as tarefas para os nós clientes e servir de porta de entrada e saída para conexão com uma rede externa, quando for o caso.

É claro que, quanto mais computadores existirem no cluster, maiores serão os custos de implementação e manutenção. Este aumento não se deve apenas ao preço dos computadores, mas também pelos equipamentos necessários para a construção de um cluster, como por exemplo switches, cabos, hubs, nobreaks, etc. Mesmo assim, os custos costumam ser menores do que a aquisição e/ou manutenção de supercomputadores, sendo que, em alguns casos, o processamento costuma ser até mais rápido.

Os clusters tem se mostrado bastante úteis, principalmente para aplicações que exijam processamento pesado como as de previsão do tempo e de abalos sísmicos, ou para aplicações que não podem parar de funcionar ou quando não podem ocorrer perda de dados como a dos sistemas de bancos.

O princípio de funcionamento de um cluster é simples: o servidor divide as tarefas e as distribui através do switch para os clientes. Em seguida, cada cliente recebe as mensagens e um conjunto de dados a serem processados, quando estes processamentos estiverem concluídos, os resultados são enviados novamente para o servidor, que os agrupam

e finaliza o processamento.

A troca de mensagens entre servidor e clientes é realizada por meio de uma biblioteca de comunicação, as mais usuais são a PVM e a MPI. A escolha da biblioteca é determinada pelo tipo de dado a ser transmitido, confiabilidade, desempenho e número de clientes. O uso de bibliotecas de comunicação é importante pois visa minimizar os problemas decorrentes da necessidade de sincronização, uma vez que as tarefas executadas em paralelo aguardam a finalização mútua para poderem então coordenar os resultados ou trocarem dados e reiniciar novas tarefas em paralelo.

Segundo [16], existem diferentes tipos de estruturas utilizadas para implementar o processamento paralelo bem como diferentes clusters. O processamento paralelo pode ser implementado utilizando: (1) Swar (*Simd Within a Register*), o qual consiste em utilizar as instruções MMX (*MultiMedia eXtensions*) disponibilizadas nos processadores para realizar tarefas em paralelo, e neste caso o processamento paralelo pode ser realizado em uma máquina com um único processador; (2) com SMP (Symetric Multi Processor), isto é, por meio de computadores com mais de um processador com as mesmas características, daí vem sua denominação; (3) com cluster de uma estação de trabalho que é um conjunto de computadores completos (com teclado, monitor, mouse), conectados em rede, e que cumprem duas funções; o uso diário, com diversos tipos de programas como processadores de texto e planilhas, e o uso para processamento paralelo pesado no final do dia e/ou nos fins de semana; (4) cluster para balanceamento de carga, no qual a distribuição de processamento aos nós do cluster é equilibrada; (5) em um cluster com MOSIX, o que trata de uma extensão para Linux (ou sistemas baseados em Unix) de um sistema de cluster que trabalha como se fosse um único supercomputador, por meio de conceitos de Distribuição de Processos e Balanceamento de Carga; (6) com cluster *Beowulf*, sendo uma tecnologia de cluster que agrupa computadores trabalhando no sistema GNU/Linux para formar um supercomputador virtual via processamento paralelo (distribuído).

Será utilizado neste trabalho este último tipo de estrutura para implementar o processamento paralelo, isto é, o cluster *Beowulf*. A seguir, esta tecnologia será apresentada de maneira mais detalhada.

3.2 Clusters *Beowulf*

Os Clusters *Beowulf* são construídos a partir de sistemas de computadores pessoais (PC) disponíveis no mercado, por este motivo tem se tornado uma escolha alternativa para a construção de sistemas de computação paralela de alta performance. O rápido avanço dos mi-

croprocessadores, das redes de interconexões de alta velocidade e das tecnologias de outros componentes têm facilitado muitas implementações bem sucedidas neste tipo de cluster.

Em seguida, será apresentado algumas definições e curiosidades sobre clusters *Beowulf*. Partes do texto desta seção 3.2 é uma tradução de [28].

O conceito de clusters *Beowulf* originou no *Center of Excellence in Space Data and Information Sciences* (CESDIS), localizado na NASA Goddard Space Flight Center, em Maryland. O primeiro cluster *Beowulf* foi desenvolvido em 1994 por Thomas Sterling e Don Becker, pesquisadores do CESDIS, para utilizá-lo em um projeto de Ciências Espaciais e Terrestres (ESS) em conjunto com o grupo de Computação e Comunicação de Alta Performance (HPCC) do qual os pesquisadores faziam parte. O protótipo inicial era um cluster de 16 processadores DX4 ligados por dois canais Ethernet acoplados (*Ethernet bonding*). A máquina foi um sucesso instantâneo e esta ideia rapidamente se espalhou pelos meios acadêmicos, pela NASA e por outras comunidades de pesquisa.

O objetivo da construção de um cluster *Beowulf* é de criar um sistema de computação paralela com custo-benefício para atender à sociedade de consumo de massa visto que os componentes podem ser facilmente encontrados à venda, satisfazendo, assim, os requisitos computacionais específicos da comunidade científica.

Beowulf é o nome de um herói escandinavo mitológico, conhecido através de um poema anglo-saxão medieval. Este herói é descrito no poema com uma força descomunal e como um homem de grande coragem e fortaleza, que lhe permitiu realizar grandes façanhas tanto na guerra como na batalha contra seres fantásticos. Neste conto épico, *Beowulf* salva o reino dinamarquês de Hrothgar de dois monstros antropófagos (Grendel e sua mãe) e de um dragão, matando cada um deles, mas termina morrendo pela gravidade das feridas da luta com tal dragão.

Beowulf é utilizado atualmente como uma metáfora para uma nova estratégia de computação de alta performance, que explora tecnologias do mercado de consumo popular para economizar tempo e dinheiro impostos pela supercomputação, liberando as pessoas para se dedicarem às suas próprias áreas de conhecimento. Ironicamente, a construção de um cluster *Beowulf* é tão divertido que os cientistas e pesquisadores ansiosamente arregaçaram as mangas para realizar as muitas tarefas tediosas envolvidas, pelo menos, na primeira vez que construíram um.

Após o sucesso do primeiro cluster *Beowulf*, vários outros foram construídos pelo CESDIS usando várias gerações e famílias de processadores e interconexões de rede. Durante o Supercomputing '96, uma conferência de supercomputação patrocinada pela *Association for Com-*

puting Machinery (ACM) e pelo *Institute of Electrical and Electronics Engineers* (IEEE®), ambos da NASA e do Departamento de Energia dos EUA, apresentou-se um cluster custando menos de $ 50.000 capaz de alcançar mais de um gigaflop por segundo mantendo a performance.

Com o rápido avanço e crescente disponibilidade de tecnologias de microprocessadores, de rede de interconexões de alta velocidade, e de outros componentes relacionados, os clusters *Beowulf* tornaram-se a escolha mais comum para a construção de computação paralela de alta performane.

Muitos supercomputadores comerciais usam os mesmos processadores, memórias e chips controladores que são utilizados pelos clusters *Beowulf*. Entretanto, clusters *Beowulf* utilizam somente componentes do mercado de consumo de massa que não estão sujeitos a atrasos devido a custos de peças personalizadas e de projetos exclusivos.

Um cluster *Beowulf* utiliza uma arquitetura multicomputacional, conforme apresentado na Fig. 3.1. Ele possui um sistema de computação paralela, que geralmente é composto por: um ou mais nós mestres; de um ou mais nós de computação, ou de nós clientes, interconectados através de uma rede de interconexões amplamente disponível. Todos os nós em um cluster *Beowulf* são sistemas facilmente encontrados no mercado como PCs, estações de trabalho ou servidores, em execução em softwares conhecidos, tais como o Linux, o Windows e macOS.

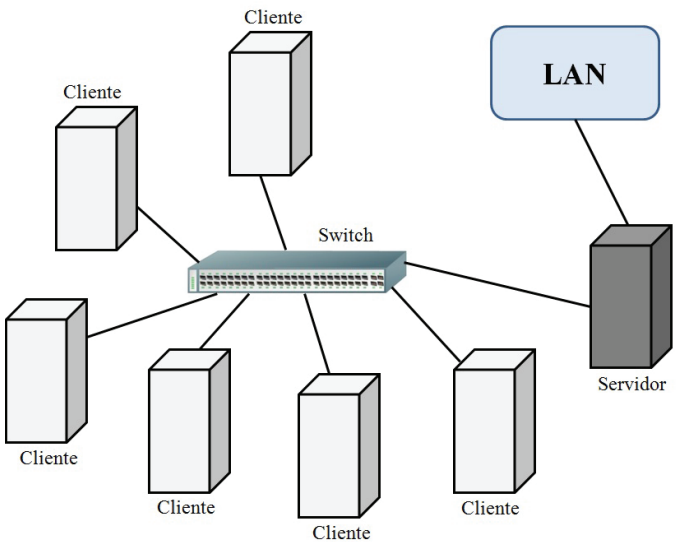


Figura 3.1: Visão Lógica de uma Classe de Cluster *Beowulf*

O nó mestre age como um servidor de Network File System (NFS), é como uma porta para o mundo exterior. Como um servidor NFS, o nó principal fornece espaço de arquivo ao usuário e outro software de sistema comum para os nós clientes via NFS. Como um gateway, o nó mestre permite aos usuários ter acesso através dele aos nós clientes. Normalmente, o nó mestre é a única máquina que também está conectada com o mundo exterior através de uma rede por meio de uma placa de interface (NIC).

A tarefa exclusiva dos nós clientes é executar tarefas em paralelo. Na maioria dos casos, portanto, os nós de computação não têm teclados, mouses, placas de vídeo ou monitores. Todo o acesso aos nós do cliente é fornecida através de conexões remotas a partir do nó mestre. Como os nós de computação não precisam acessar máquinas de fora do cluster, nem máquinas de fora do cluster precisam acessar diretamente os nós de computação, normalmente os nós de computação usam endereços de IP privados, como as faixas de endereços 10.0.0.0 / 8 ou 192.168.0.0/16.

Do ponto de vista de um usuário, um cluster *Beowulf* aparece como um Processador Massivamente Paralelo (PMP) do sistema. Os métodos mais comuns de usar o sistema são: acessar o nó mestre diretamente ou através de Telnet; ou obter um login remoto a partir de estações de trabalho pessoais. Após acessar o nó mestre, os usuários podem preparar e compilar suas aplicações paralelas, e também gerar trabalho em um número desejado de nós de computação em cluster.

As aplicações devem ser escritas em paralelo e usar um programa modelo de transmissão de mensagens. Trabalhos de aplicação paralela são gerados em nós de computação, que trabalham colaborativamente até terminarem a aplicação. Durante a execução, os nós de computação usam um padrão de transmissão de mensagens middleware, como o *Message Passing Interface* (MPI) e *Parallel Virtual Machine* (PVM), para trocar informações.

Clusters *Beowulf* oferecem uma série de benefícios específicos:

Custo-benefício: Uma das principais vantagens de uma cluster *Beowulf* é o seu custo-efetividade. Clusters *Beowulf* são construídos com componentes que são relativamente baratos e amplamente disponíveis no mercado.

Mantém o passo com as tecnologias: Já que os clusters *Beowulf* utilizam apenas os componentes de mercado de massa, é fácil de empregar as últimas tecnologias para manter o cluster no estado da arte.

Configuração flexível: Os usuários podem personalizar uma configuração que lhe é viável e repartir o orçamento com sabedoria para satisfazer as exigências de desempenho de seus aplicativos. Por exemplo, aplicações em paralelo de grão fino (que trocam muitas mensagens

entre os processadores frequentemente) podem motivar os usuários a atribuir uma parcela maior do seu orçamento para interconexões de alta velocidade.

Escalabilidade: Quando o poder de processamento requer aumento, a performance e o tamanho de um cluster *Beowulf* pode ser facilmente expandido pela adição de mais nós de computação.

Alta disponibilidade: Cada nó de computação de um cluster *Beowulf* é uma máquina individual. A falha de um nó de computação não afetará os outros nós ou a disponibilidade de todo o cluster.

Compatibilidade e portabilidade: Graças à padronização e ampla disponibilidade de interfaces de passagem de mensagens, como MPI e PVM, a maioria das aplicações paralelas usam estes middlewares padrões. É por isso que as classes de clusters *Beowulf* estão substituindo rapidamente os caros computadores paralelos do mercado de baixo porte para o mercado de médio porte da computação de alto desempenho.

3.3 Métricas de Desempenho

Medir o desempenho de um processamento em paralelo é encontrar uma forma de caracterizar o comportamento do sistema para então interpretar de forma correta os resultados, e esta interpretação dependerá fortemente da escolha das métricas de aferição de desempenho. Aqui, os termos performance ou desempenho de um sistema se referirão ao tempo total de execução de um programa, desta forma quanto menor for o tempo de execução, maior será o desempenho. Em outras palavras pode-se dizer que desempenho é a capacidade de reduzir o tempo de resolução do problema à medida que os recursos computacionais aumentam.

A aferição do desempenho é muito importante pois no processo de desenvolvimento de um sistema computacional um dos principais objetivos é obter a maior eficiência com o menor custo possível.

Pode-se dividir as métricas de desempenho em duas classes: (a) nas métricas de desempenho para processadores, as quais permitem avaliar a performance de um processador tendo por base a velocidade/número de operações que este consegue realizar num determinado espaço temporal; (b) métricas de desempenho para aplicações paralelas as quais permitem avaliar a performance de uma aplicação paralela tendo por base a comparação entre a execução com múltiplos processadores e a execução com um só processador. Este último caso será considerado em seguida.

O tempo total de execução de um programa pode ser definido como o tempo que decorre desde que o primeiro processador inicia a execução

até o último processador terminar e pode ser decomposto no tempo de computação, de comunicação e no tempo ocioso, ou seja:

$$T_{exec} = T_{comp} + T_{comu} + T_{ocioso},$$

sendo que, o tempo de computação é o tempo gasto no processamento excluindo-se o tempo de comunicação e o tempo ocioso; o tempo ocioso é o período que um processador fica sem tarefas, talvez, por exemplo, esperando por algum dado de outro processador, o que pode ser minimizado com uma distribuição de carga adequada ou sobrepondo a computação com a comunicação e finalmente, o tempo de comunicação é o tempo que o algoritmo gasta para enviar e receber mensagens.

De acordo com [45], "desempenho em programação paralela é uma questão multifacetada. Além do tempo de execução e escalabilidade é importante considerar os mecanismos pelos quais os dados são gerados, armazenados, transmitidos na rede, transferidos de/para o disco e transportados nos diferentes estágios de computação. As principais métricas encontradas nesta área são: tempo de execução e throughput (citados anteriormente), requisitos de memória, requisitos de entrada/saída, latência (tempo que uma unidade de informação leva para chegar ao seu destino através de um meio de comunicação), custos de projeto, de implementação, verificação de hardware, de potencial de reuso e de portabilidade". A comunicação e a sincronização entre diferentes subtarefas é tipicamente uma das maiores barreiras para atingir um grande desempenho em programas paralelos.

Existem várias métricas que permitem medir e/ou avaliar o desempenho de uma aplicação paralela. Aqui serão apresentadas as mais conhecidas.

As métricas *Speedup* (S_p) e Eficiência (E_p) são equações das frações de tempo que os processadores passam realizando trabalho útil, caracterizando a efetividade de se utilizar paralelismo em relação a uma versão sequencial.

Sejam p o número de processadores, T_1 o tempo em segundos de execução do programa em um processador e T_p o tempo em segundos de execução do programa em paralelo com p processadores. Segundo [47], o *Speedup*, dado pela Eq. (3.3.1), é a razão entre o tempo de execução sequencial e o tempo de execução em paralelo e medirá o grau de desempenho do processamento, ou seja, o ganho de velocidade de processamento.

$$S_p = \frac{T_1}{T_p} \tag{3.3.1}$$

O *speedup linear* ou *speedup ideal* é dado por $S_{p_l} = p$, isto é, o *speedup* ideal significa que o tempo de execução do algoritmo sequencial em um processador é igual ao tempo de execução do algoritmo em paralelo

com p processadores multiplicado por p, em outras palavras, $T_1 = pT_p$, o que indica que toda a capacidade computacional disponível no sistema foi utilizada para ganhar velocidade na execução do algoritmo. Mas tal valor é quase impossível de se alcançar, pois para que isto aconteça o algoritmo precisa ser altamente paralelizável e a comunicação entre os processadores deve ser mínima.

De acordo com [47], alguns fatores que podem gerar sobrecargas no sistema e portanto impedir que o sistema atinja o valor do *speedup* ideal são o excesso de comunicação entre os processadores, o nível de paralelismo utilizado, distribuição de carga e a existência de partes do código executável que sejam estritamente sequenciais.

Se a sobrecarga da paralelização for muito alta então $T_1 < T_p$ donde $S_p < 1$. Esta situação indesejável é chamada de *slowdown*. O comportamento comum do *speedup* é chamado de sublinear e ocorre quando $1 < S_p < p$. O *speedup ideal* é dado por $S_{p_l} = p$ e *speedup* supralinear ocorre quando $S_p > p$, não é uma situação comum mas é possível de acontecer.

Por outro lado, a Eficiência, dada pela Eq. (3.3.2), é a razão entre o grau de desempenho e a quantidade de recursos computacionais e medirá o grau de aproveitamento destes recursos disponíveis.

$$E_p = \frac{S_p}{p} = \frac{T_1}{pT_p} \tag{3.3.2}$$

Observe que algoritmos com *speedup ideal* ou algoritmos sendo executados em um único processador tem eficiência igual a um ou 100% porém, a medida que o número de processadores vai aumentando a eficiência se aproxima de zero visto que $\lim_{p\to\infty} \frac{T_1}{pT_p} = 0$.

De forma similar à classificação do *speedup* pode-se classificar a eficiência. Se $E_p < 1$ o sistema encontra-se na situação de slowdown; se $E_p < 1$, então é sublinear, se $E_p = 1$ é linear e se $E_p > 1$ então é supralinear.

Pelo que foi visto pode-se concluir que a eficiência de uma aplicação é uma função decrescente em relação ao número de processadores, Fig. 3.2, ou é uma função crescente em relação ao tamanho do problema, Fig. 3.3.

Uma aplicação é dita escalável quando demonstra a capacidade de manter a mesma eficiência à medida que o número de processadores e a complexidade do problema aumentam proporcionalmente.

É importante lembrar que numa aplicação sempre existe uma parte do algoritmo que não pode ser paralelizada, ou seja, sempre existem partes do algoritmo que são estritamente sequenciais. Afim de se obter um limite máximo do valor do *speedup* que pode-se encontrar ao

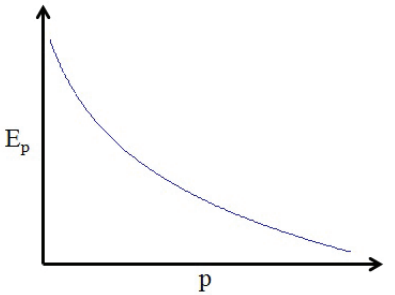


Figura 3.2: Eficiência para tamanho de problema fixo.

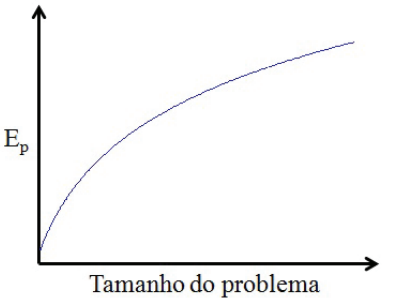


Figura 3.3: Eficiência com número de processadores fixo.

resolver um determinado problema com p processadores será considerado a Lei de Amdahl. Seja s a fração do código que é estritamente sequencial assim, $(1 - s)$ será a parte suscetível de ser paralelizada. Observe que na Lei de Amdahl parte do tempo de execução sequencial é utilizado para estimar o speedup máximo possível de ser alcançado em múltiplos processadores. As Figs. 3.4 e 3.5 ilustram a relação entre tempo de execução e número de processadores na visão de Amdahl, na qual assume-se que s seja constante para qualquer quantidade de processadores.

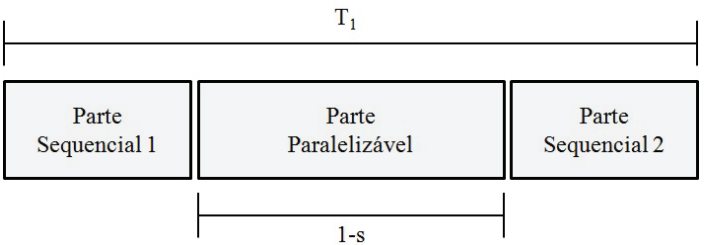


Figura 3.4: Lei de Amdahl: tempo de execução em um único processador.

Além disso, a computação realizada por uma aplicação paralela pode ser dividida em computações que só podem ser realizadas sequencialmente, C_{seq}, em computações que podem ser realizadas em paralelo, C_{par}, e em computações de comunicação/sincronização/iniciação, C_{com}. Desta forma o *speedup* pode ser reescrito da seguinte maneira:

$$S_p = \frac{T_1}{T_p} = \frac{C_{seq} + C_{par}}{C_{seq} + \frac{C_{par}}{p} + C_{com}} \tag{3.3.3}$$

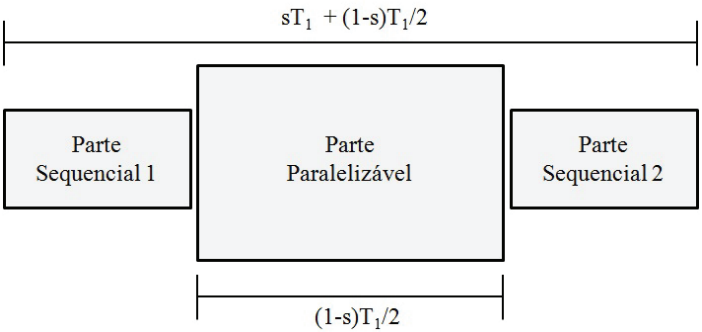


Figura 3.5: Lei de Amdahl: tempo de execução em dois processadores.

Como, em aplicações paralelas, $C_{com} \geq 0$ segue que $C_{seq} + \frac{C_{par}}{p} \leq C_{seq} + \frac{C_{par}}{p} + C_{com}$, donde

$$\frac{C_{seq} + C_{par}}{C_{seq} + \frac{C_{par}}{p}} \geq \frac{C_{seq} + C_{par}}{C_{seq} + \frac{C_{par}}{p} + C_{com}} = S_p \quad \rightarrow \quad S_p \leq \frac{C_{seq} + C_{par}}{C_{seq} + \frac{C_{par}}{p}} \tag{3.3.4}$$

Como s representa a fração da computação que só pode ser realizada sequencialmente segue que $s = \frac{C_{seq}}{C_{seq} + \frac{C_{par}}{p}}$, substituindo na Eq. (3.3.4) e simplificando o resultado chega-se na Lei de Amdahl:

$$S_p \leq \frac{1}{s - \frac{1-s}{p}} \qquad 0 \leq s \leq 1 \tag{3.3.5}$$

Pode-se também utilizar a lei de Amdahl para determinar o limite máximo de *speedup* que uma determinada aplicação poderá alcançar independentemente do número de processadores a utilizar, isto é,

$$\lim_{p \to \infty} \frac{1}{s - \frac{1-s}{p}} = \frac{1}{s} \tag{3.3.6}$$

Observe que a lei de Amdahl ignora o custo das operações de comunicação/sincronização associadas à introdução de paralelismo numa aplicação. Por esse motivo, a lei de Amdahl pode resultar em predições pouco realísticas para determinados problemas. Mais ainda, à medida que se aumenta o parâmetro p o *Speedup*, segundo a Lei de Amdahl, estará limitado a $1/s$, o que tornou-se um incômodo para os fabricantes de máquinas de grande porte. Por este motivo, uma alternativa para se medir o desempenho do sistema é a utilização da Lei de Gustafson-Barsis. Esta lei parte do princípio de que todo problema

suficientemente grande pode ser eficientemente paralelizável, diferentemente da Lei de Amdahl no qual a fração que pode ser executada em paralelo e a quantidade de processadores dependem um do outro. Mais ainda, como será visto, pela Lei de Gustafson-Barsis o *Speedup* cresce indefinidamente.

Afim de se estruturar a Lei de Gustafson-Barsis considere novamente a Eq. (3.3.4). Como

$$s = \frac{C_{seq}}{C_{seq} + \frac{C_{par}}{p}} \quad \text{segue que} \quad 1 - s = \frac{\frac{C_{par}}{p}}{C_{seq} + \frac{C_{par}}{p}} \tag{3.3.7}$$

Assim, tem-se que

$$C_{seq} = s\left(C_{seq} + \frac{C_{par}}{p}\right) \quad \text{e} \quad C_{par} = p(1-s)\left(C_{seq} + \frac{C_{par}}{p}\right) \tag{3.3.8}$$

Substituindo na Eq. (3.3.4) segue

$$\begin{aligned} S_p &\leq \frac{C_{seq}+C_{par}}{C_{seq}+\frac{C_{par}}{p}} \\ &= \frac{s\left(C_{seq}+\frac{C_{par}}{p}\right)+p(1-s)\left(C_{seq}+\frac{C_{par}}{p}\right)}{C_{seq}+\frac{C_{par}}{p}} \\ &= \frac{(s+p(1-s))\left(C_{seq}+\frac{C_{par}}{p}\right)}{\left(C_{seq}+\frac{C_{par}}{p}\right)} \end{aligned} \tag{3.3.9}$$

Logo, a Lei de Gustafson-Barsis é dada pela inequação

$$S_p \leq p + s(1-p) \tag{3.3.10}$$

Como analisado, a Lei de Gustafson-Barsis parte do tempo de execução em paralelo para estimar o *speedup* máximo comparado com a execução sequencial. Além disso, *speedup* máximo segundo a Lei de Gustafson-Barsis converge para o infinito quando o número de processadores cresce indefinidamente.

Mas, assim como na lei de Amdahl, a Lei de de Gustafson-Barsis também tem limitações, ao usar o tempo de execução em paralelo como o ponto de partida, em lugar do tempo de execução sequencial, a Lei de Gustafson-Barsis assume que a execução com um só processador é no pior dos casos p vezes mais lenta que a execução com p processadores o que pode não ser verdade.

Finalmente, será abordado aqui a Métrica de de Karp-Flatt. Seja $\alpha = \frac{C_{seq}}{T_1}$ a razão sequencial determinada experimentalmente numa computação paralela. Assim, $C_{seq} = \alpha T_1$ e lembrando que se pode escrever

$T_1 = C_{seq} + C_{par}/p$ segue que $C_{par} = T_1 - C_{seq} = T_1 - \alpha T_1 = (1-\alpha)T_1$. Considerando que C_{com} é desprezável em $T_p = C_{seq} + C_{par}/p + C_{com}$ vem que

$$T_p = \alpha T_1 + \frac{(1-\alpha)T_1}{p} \tag{3.3.11}$$

Mas, $S_p = \frac{T_1}{T_p} \quad \rightarrow \quad T_1 = S_P T_p$. Substituindo na Eq. (3.3.11) segue que:

$$T_p = \alpha S_p T_p + \frac{(1-\alpha)S_p T_P}{p}, \tag{3.3.12}$$

simplificando T_p em ambos os lados da equação obtem-se

$$1 = \alpha S_p + \frac{(1-\alpha)S_p}{p} = S_p\left(\alpha + \frac{1-\alpha}{p}\right)$$

daí

$$\frac{1}{S_p} = \alpha + \frac{1-\alpha}{p} = \alpha\left(1 - \frac{1}{p}\right) + \frac{1}{p}$$

ou seja,

$$\frac{1}{S_p} = \alpha\left(1 - \frac{1}{p}\right) + \frac{1}{p} \tag{3.3.13}$$

Isolando α chega-se a Métrica de de Karp-Flatt:

$$\alpha = \frac{\frac{1}{S_p} - \frac{1}{p}}{1 - \frac{1}{p}} \tag{3.3.14}$$

Observe que, por definição, α é um valor constante que não depende do número de processadores, mas pela métrica de Karp-Flatt α é uma função do número de processadores. Como a eficiência duma aplicação é uma função decrescente do número de processadores, a métrica de Karp-Flatt permite-nos determinar qual a importância da componente C_{com} nesse decréscimo.

Se os valores de α forem constantes à medida que o número de processadores aumenta isso significa que a componente C_{com} é também constante. Logo, o decréscimo da eficiência é devido à existência de pouco paralelismo no problema.

Por outro lado, se os valores de α aumentarem à medida que o número de processadores aumentam isso significa que o decréscimo é devido à componente C_{com}, ou seja, à existência de custos excessivos associados à computação em paralelo (custos de comunicação, sincronização e/ou iniciação da computação).

Há ainda várias métricas para se avaliar o desempenho de um processo em paralelo que não serão analisadas aqui. Para mais considerações de métricas de desempenho em computação paralela pode-se citar os livros [29] e [52].

A fim de ilustrar o uso das métricas *speedup* e a eficiência, suponha que o tempo em segundos de execução em um mesmo problema de 1, 2, 4, 8 e 16 processadores sejam respectivamente, 1500, 850, 470, 260 e 150. A Tab. 3.1 apresenta os valores obtidos do *speedup* e da eficiência para cada caso. Note que estas métricas são grandezas adimensionais.

No exemplo ilustrativo apresentado na Tab. 3.1 o *speedup* em todos os casos é sublinear, isto é, $1 < S_p < p$. Como previsto na Lei de Gustafson-Barsis o *speedup* aumenta quando o número de processadores cresce. A eficiência neste exemplo também é sublinear e decrescente com relação ao número de processadores.

Tabela 3.1: Ilustração sobre as métricas *Speedup* e Eficiência

	1 CPU	2CPUs	4 CPUs	8 CPUs	16 CPUs
T_p	1500	850	470	260	150
S_p	$S_1 = \frac{1500}{1500} = 1$	$S_2 = \frac{1500}{850} = 1,76$	$S_4 = \frac{1500}{470} = 3,19$	$S_8 = \frac{1500}{260} = 5,77$	$S_{16} = \frac{1500}{150} = 10$
E_p	$E_1 = \frac{1}{1} = 1$	$E_2 = \frac{1,76}{2} = 0,88$	$E_4 = \frac{3,19}{4} = 0,80$	$E_8 = \frac{5,77}{8} = 0,72$	$E_{16} = \frac{10}{16} = 0,62$

Capítulo 4

Simulações numéricas

Este capítulo é voltado para as simulações numéricas realizadas durante o estudo a fim de comprovar a eficiência do algoritmo Evolução Diferencial Melhorada implementado em processamento paralelo. O algoritmo da EDMP foi desenvolvido em C++ pelos autores deste livro.

4.1 Aplicações Simples em Processamento Paralelo

Várias funções testes de otimização, irrestritas e restritas, foram utilizadas a fim de se encontrar possíveis erros de implementação e verificar a capacidade de otimização do código computacional Evolução Diferencial Melhorada implementado em paralelo. Todos os problemas usados para validação do código foram consultados no site [40]. As simulações foram realizadas com 1, 2, 4, 8, 16 e 32 processadores e os resultados serão apresentados a seguir.

As simulações numéricas desta seção 4.1 foram desenvolvidas no cluster do Laboratório de Mecânica dos Fluidos e Análise Numérica da FACIP cujos 36 processadores homogêneos são AMD AM3 Phenom II FX-6100 3.3 GHz.

Em todos os problemas foram usados 1600 indivíduos na população inicial, 200 iterações do algoritmo da ED em cada execução do algoritmo da EDMP, taxa de pertubação F=0,8 e probabilidade de cruzamento CR=0,6.

Vale ressaltar que o algoritmo da EDMP possui um laço externo para executar 20 rodadas do algoritmo e se obter a média e o desvio padrão das métricas. As múltiplas rodadas acontecem tanto na versão sequencial do algoritmo, usando um processador, quanto nas versões em paralelo, com vários processadores.

Observe que a escolha do número de indivíduos está relacionado ao número de processadores visto que a quantidade de indivíduos deve ser divisível pela quantidade de processadores que será utilizada nas

Tabela 4.1: Função teste de Beale.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$x_{\text{ótimo}}$	$f(x_{\text{ótimo}})$
1	51,5	1	1	(3; 0,5) $(3; 0, 5)^*$ 0^{**}	$f = 0$ $\overline{f} = 0^*$ $\sigma = 0^{**}$
2	24,1	2,14	1,07	(3; 0,5) $(3; 0, 5)^*$ 0^{**}	$f = 0$ $\overline{f} = 0^*$ $\sigma = 0^{**}$
4	12,5	4,12	1,03	(3; 0,5) $(3; 0, 5)^*$ 0^{**}	$f = 0$ $\overline{f} = 0^*$ $\sigma = 0^{**}$
8	6,8	7,58	0,95	(3; 0,5) $(3; 0, 5)^*$ 0^{**}	$f = 0$ $\overline{f} = 0^*$ $\sigma = 0^{**}$
16	2,8	18,39	1,15	(3; 0,5) $(3; 0, 5)^*$ 0^{**}	$f = 0$ $\overline{f} = 0^*$ $\sigma = 0^{**}$
32	2,26	22,79	0,71	(3; 0,5) $(3; 0, 5)^*$ 0^{**}	$f = 0$ $\overline{f} = 0^*$ $\sigma = 0^{**}$

* Valores médios e ** desvios padrões.

simulações. Foi adotado 1600 indivíduos pois este número é divisível por 1, 2, 4, 8, 16 e 32 processadores. Assim, no caso das simulações com 32 processadores haverá 50 indivíduos em cada processador.

4.1.1 Função de Beale

Seja o problema de otimização irrestrito dado por:

$$min\ f(x) = (1,5 - x_1 + x_1x_2)^2 + (2,25 - x_1 + x_1x_2^2)^2 + (2,625 - x_1 + x_1x_2^3)^2 \tag{4.1.1}$$

Considerando os limites laterais das variáveis: $-4,5 \leq x_i \leq 4,5$ i=1,2. Sabe-se que $x^* = (3; 0, 5)$ e $f(x^*) = 0$. Os resultados obtidos são apresentados na Tab. 4.1.

Observe que o tempo de processamento sequencial é 18 vezes maior que o tempo necessário para 16 processadores resolver o mesmo problema e que em todos os casos a solução ótima foi encontrada. No entanto, a diferença do tempo de processamento entre 16 e 32 processadores é de décimos de segundos. Um dos motivos desta pequena diferença de tempo se deve ao aumento da comunicação que resulta numa computação menos eficiente e, portanto, mais lenta.

O speedup indica quantas vezes o programa em paralelo é mais rápido que a versão sequencial para executar uma dada tarefa. Se $S_p > 1$ a versão paralela reduziu o tempo de execução, isto é, ficou mais rápido que a sequencial. Se $S_p < 1$ a versão paralela aumentou o tempo de execução, ou seja, ficou mais lenta que a sequencial.

Nas simulações com a função de Beale a versão paralela é mais rápida que a versão sequencial já que em todos os casos tem-se $S_p > 1$. No entanto, o valor do speedup com mais de 16 processadores tende a estagnar e, desde que se mantenha o problema fixo, o tempo de execução não diminuirá muito com o aumento do número de processadores.

Visto que para 2, 4 e 16 processadores o valor do speedup é maior que o número de processadores, nestes casos o speedup mostra-se super linear. Para 8 e 32 processadores o speedup é sublinear e para 1 processador ele é linear.

Normalmente as unidades ativas ficam parte do tempo esperando por resultados vizinhos assim, o objetivo da métrica eficiência é indicar a taxa de utilização média das unidades ativas, isto é, estima-se o quão bem os processadores estão sendo utilizados para resolver o problema.

A eficiência ideal é aquela em que em 100% do tempo cada unidade está ativa. Valores acima de 1 indicam eficiência super linear como nos casos das simulações com 2, 4 e 16 processadores apresentados na Tab. 4.1. Já nas simulações com 8 e 32 processadores a eficiência mostrou-se sublinear.

4.1.2 Função de Michalewicz

Neste teste o objetivo é minimizar a função de Michalewicz para dez variáveis, isto é, n=10, cuja função objetivo é dada por:

$$min\ f(x) = -\sum_{i=1}^{n} sen(x_i)\left(sen\left(\frac{i * x_i^2}{\pi}\right)\right)^{2n} \qquad (4.1.2)$$

sendo $0 \leq x_i \leq \pi, \quad i = 1, \cdots, 10$ os limites laterais da função e $f(x_{\text{ótimo}}) = -9,66015$.

Os valores ótimos podem ser observados na Tab. 4.2. Para simplificar, considerando que n=10, foram omitidos os valores encontrados relacionados ao $x_{\text{ótimo}}$.

Neste teste, houve uma diminuição significativa do tempo de processamento em paralelo quando comparado ao sequencial. Por exemplo, ao utilizar 2 processadores na execução do programa o tempo de processamento reduziu pela metade. O que representa um excelente ganho em tempo de execução.

Embora os desvios padrões do valor ótimo da função objetivo tenha aumentado com o aumento do número de processadores, fato este jus-

Tabela 4.2: Função teste de Michalewicz.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	100,4	1	1	$f = -9,66015$ $\overline{f} = -9,60704^{*}$ $\sigma = 0,10346^{**}$
2	46,27	2,17	1,08	$f = -9,66015$ $\overline{f} = -9,59844^{*}$ $\sigma = 0,16887^{**}$
4	23,77	4,22	1,05	$f = -9,66015$ $\overline{f} = -9,57757^{*}$ $\sigma = 0,20428^{**}$
8	12,82	7,83	0,96	$f = -9,66015$ $\overline{f} = -9,56889^{*}$ $\sigma = 0,24053^{**}$
16	6,35	15,81	0,99	$f = -9,66015$ $\overline{f} = -9,54849^{*}$ $\sigma = 0,25475^{**}$
32	4,14	24,25	0,76	$f = -9,66015$ $\overline{f} = -9,53553^{*}$ $\sigma = 0,27983^{**}$

* Valores médios e ** desvios padrões.

tificado pelo grande número de avaliações da função objetivo, em todos os casos a solução ótima encontrada corresponde à analítica.

Pode-se observar na Tab. 4.2 que nas simulações com 2 e 4 processadores o speedup e a eficiência apresentaram valores super lineares.

4.1.3 Função de Levy

A função de Levy é dada pela seguinte equação:

$$min\ f(x) = sen^2(\pi z_1) + \sum_{i=1}^{n-1} \left[(z_i - 1)^2(1 + 10sen^2(\pi z_i + 1))\right] + (z_n - 1)^2\left(1 + sen^2(2\pi z_n)\right) \tag{4.1.3}$$

onde $z_i = 1 + \frac{x_i - 1}{4}$ e $-10 \leq x_i \leq 10,\ \ i = 1, \cdots, n$. O problema foi resolvido para o caso n=30. Sabe-se que $x_{\text{ótimo}} = (1, \cdots, 1)$ e $f(x_{\text{ótimo}}) = 0$. A Tab. 4.3 resume os resultados encontrados para $f(x_{\text{ótimo}})$.

Novamente, a redução do tempo de execução do algoritmo em paralelo é muito significativa. Para a função de Levy o tempo de processamento sequencial é 25 vezes maior que o tempo necessário para 32 processadores resolver o problema. A solução ótima é encontrada em todos os casos e os devios padrões dos valores ótimos da função objetivo são muito baixos.

Tabela 4.3: Função teste de Levy.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	218,67	1	1	$f = 0$ $\overline{f} = 0,00155^{*}$ $\sigma = 0,01529^{**}$
2	95,9	2,28	1,14	$f = 0$ $\overline{f} = 0,00107^{*}$ $\sigma = 0,01066^{**}$
4	43,6	5,01	1,25	$f = 0$ $\overline{f} = 0,00132^{*}$ $\sigma = 0,01327^{**}$
8	20,52	10,66	1,33	$f = 0$ $\overline{f} = 0,00117^{*}$ $\sigma = 0,01164^{**}$
16	11,41	19,16	1,2	$f = 0$ $\overline{f} = 0,00118^{*}$ $\sigma = 0,01262^{**}$
32	8,72	25,08	0,78	$f = 0$ $\overline{f} = 0,00095^{*}$ $\sigma = 0,00978^{**}$

* Valores médios e ** desvios padrões.

Tabela 4.4: Função teste de Shubert.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	66,85	1	1	$f = -186,7309$ $\overline{f} = -186,7309^{*}$ $\sigma = 0,00002^{**}$
2	32,15	2,08	1,04	$f = -186,7309$ $\overline{f} = -186,7309^{*}$ $\sigma = 0,00004^{**}$
4	16,91	3,95	0,98	$f = -186.7309$ $\overline{f} = -186,7308^{*}$ $\sigma = 0,00027^{**}$
8	8,9	7,51	0,93	$f = -186.7309$ $\overline{f} = -186,7308^{*}$ $\sigma = 0,00024^{**}$
16	4,23	15,8	0,99	$f = -186.7309$ $\overline{f} = -186,7308^{*}$ $\sigma = 0,0009^{**}$
32	3,01	22,21	0,69	$f = -186.7309$ $\overline{f} = -186,7306^{*}$ $\sigma = 0,0035^{**}$

* Valores médios e ** desvios padrões.

Pode-se observar na Tab. 4.3 que com os testes com a função de Levy apenas a simulação com 32 processadores não apresentou valores super lineares para o speedup e a eficiência.

4.1.4 Função de Shubert

A função de Shubert é uma função de duas variáveis e por ser multimodal, possuindo 18 mínimos globais, torna-se uma excelente opção para testar algoritmos de otimização. Seu gráfico está ilustrado na Fig. 4.1 (a) e os valores ótimos podem ser vistos na Tab. 4.4. A função de Shubert é escrita como:

$$min\ f(x) = \left(\sum_{i=1}^{5} icos((i+1)x_1 + i)\right) \cdot \left(\sum_{i=1}^{5} icos((i+1)x_2 + i)\right) \tag{4.1.4}$$

Tem-se que $f(x_{\text{ótimo}}) = -186,7309$. Observe que em todos os casos a solução ótima foi encontrada e que os valores ótimos do desvio padrão das funções objetivos foram muito baixos, indicando assim que mesmo dividindo a execução do problema em vários processadores a solução ótima foi obtida sem muita variância no seu valor.

Tabela 4.5: Função teste de Rastrigin.

N	T	Speedup	Eficiência	$x_{\text{ótimo}}$	$f(x_{\text{ótimo}})$
1	54,73	1	1	$(0;\ 0)$ $(0,1219 \cdot 10^{-7}; 0,6467 \cdot 10^{-7})^*$ $(0,1208 \cdot 10^{-6}; 0,6504 \cdot 10^{-6})^{**}$	$f = 0$ $\overline{f} = 0,8 \cdot 10^{-10^*}$ $\sigma = 0,8 \cdot 10^{-9^{**}}$
2	25,71	2,13	1,06	$(0;\ 0)$ $(0,6589 \cdot 10^{-7}; 0,2375 \cdot 10^{-7})^*$ $(0,6884 \cdot 10^{-6}; 0,3241 \cdot 10^{-6})^{**}$	$f = 0$ $\overline{f} = 0,11 \cdot 10^{-9^*}$ $\sigma = 0,11 \cdot 10^{-8^{**}}$
4	13,09	4,18	1,04	$(0;\ 0)$ $(0,4967 \cdot 10^{-7}; 0,287 \cdot 10^{-8})^*$ $(0,1144 \cdot 10^{-5}; 0,931 \cdot 10^{-6})^{**}$	$f = 0$ $\overline{f} = 0,43 \cdot 10^{-9^*}$ $\sigma = 0,5 \cdot 10^{-8^{**}}$
8	7,36	7,44	0,93	$(0;\ 0)$ $(0,5623 \cdot 10^{-7}; -0,837 \cdot 10^{-8})^*$ $(0,1371 \cdot 10^{-5}; 0,758 \cdot 10^{-6})^{**}$	$f = 0$ $\overline{f} = 0,48 \cdot 10^{-9^*}$ $\sigma = 0,8 \cdot 10^{-8^{**}}$
16	3,23	16,94	1,05	$(0;\ 0)$ $(0,2026 \cdot 10^{-7}; 0,2749 \cdot 10^{-7})^*$ $(0,1676 \cdot 10^{-5}; 0,1694 \cdot 10^{-5})^{**}$	$f = 0$ $\overline{f} = 0,112 \cdot 10^{-8^*}$ $\sigma = 0,16 \cdot 10^{-7^{**}}$
32	2,52	21,72	0,68	$(0;\ 0)$ $(-0,1176 \cdot 10^{-7}; 0,3829 \cdot 10^{-7})^*$ $(0,7308 \cdot 10^{-5}; 0,4985 \cdot 10^{-5})^{**}$	$f = 0$ $\overline{f} = 0,1552 \cdot 10^{-7^*}$ $\sigma = 0,355 \cdot 10^{-6^{**}}$

* Valores médios e ** desvios padrões.

Note que para as simulações com a função de Shubert apenas o caso com 2 processadores apresentou valores super lineares para o speedup e a eficiência.

4.1.5 Função de Rastrigin

Seja o mínimo da função de Rastrigin representado pela seguinte equação:

$$min\ f(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10cos(2\pi x_i)\right] \tag{4.1.5}$$

sendo que $-5,12 \leq x_i \leq 5,12, \quad i = 1, \cdots, n$. O problema foi resolvido para o caso n=2 onde $x_{\text{ótimo}} = (0;0)$ e $f(x_{\text{ótimo}}) = 0$. A Tab. 4.5 apresenta os resultados ótimos e sua representação gráfica pode ser observada na Fig. 4.1 (b). Devido ao espaçamento, na Tab. 4.5 foi abreviado "número de processadores" para N e "tempo de processamento em segundos" para T.

A solução ótima, $x_{\text{ótimo}}$, foi encontrada com no mínimo 5 casas decimais de precisão. O tempo de execução do programa para 32 processadores foi menor que 3 segundos e o desvio padrão dos valores ótimos da função objetivo foram muito baixos, comprovando assim a eficiência do algoritmo em paralelo.

Assim como ocorreu nos testes com a função de Beale, as simulações com 2, 4 e 16 processadores da função de Rastrigin apresentaram speedup e eficiência com valores super lineares, conforme exposto na Tab. 4.5.

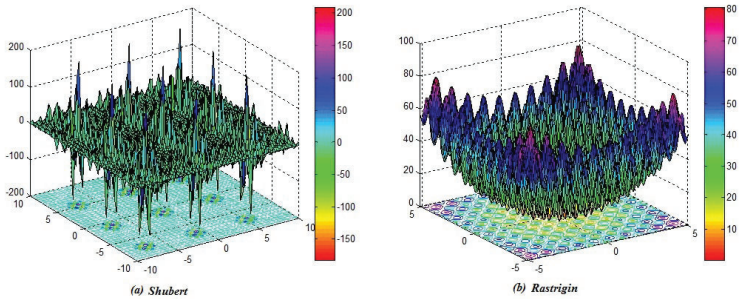


Figura 4.1: Representação gráfica das funções: (a) Shubert e (b) Rastrigin.

4.1.6 Problemas Restritos

Foram analisados dois problemas testes com restrições. Normalmente, os métodos heurísticos são desenvolvidos para problemas irrestritos. Mas no desenvolvimento do código computacional para o EDMP foram consideradas a presença de restrições utilizando o Método da Penalidade. Para estes problemas, adotou-se um fator de penalidade elevado da ordem 10^5, conforme Eqs. (1.1.3) e (1.1.4).

Problema Restrito 1: O primeiro problema restrito testado, dado pela Eq. (4.1.6), envolve treze variáveis de projeto e nove restrições representadas pelas Eqs. de (4.1.7) a (4.1.15).

$$min\ f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{4.1.6}$$

Sujeito a:

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \tag{4.1.7}$$

$$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \tag{4.1.8}$$

$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \tag{4.1.9}$$

$$g_4(x) = -8x_1 + x_{10} \leq 0 \tag{4.1.10}$$

$$g_5(x) = -8x_2 + x_{11} \leq 0 \tag{4.1.11}$$

$$g_6(x) = -8x_3 + x_{12} \leq 0 \tag{4.1.12}$$

$$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0 \tag{4.1.13}$$

$$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0 \tag{4.1.14}$$

$$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0 \tag{4.1.15}$$

Tabela 4.6: Problema restrito 1.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	91,74	1	1	$f = -15$ $\overline{f} = -14,0906^{*}$ $\sigma = 1,1327^{**}$
2	45,16	2,03	1,02	$f = -15$ $\overline{f} = -14,3453^{*}$ $\sigma = 1,0125^{**}$
4	21,58	4,25	1,06	$f = -15$ $\overline{f} = -14,5726^{*}$ $\sigma = 0,8623^{**}$
8	10,68	8,59	1,07	$f = -15$ $\overline{f} = -14,8863^{*}$ $\sigma = 0,4995^{**}$
16	4,98	18,42	1,15	$f = -15$ $\overline{f} = -14,0123^{*}$ $\sigma = 1,2119^{**}$
32	4,03	22,76	0,71	$f = -15$ $\overline{f} = -14,113^{*}$ $\sigma = 1,2325^{**}$

* Valores médios e ** desvios padrões.

A solução ótima encontrada para o problema restrito 1 é $x_{\text{ótimo}} = (1, 1, \cdots, 1, 3, 3, 3, 1)$ e $f(x_{\text{ótimo}}) = -15$. Os limites laterais das variáveis são:

$$0 \leq x_i \leq 1, \quad i = 1, \cdots, 9, 13, \quad \text{e} \quad 0 \leq x_i < 100, \quad i = 10, 11, 12$$

Na Tab. 4.6 são apresentados um resumo dos valores encontrados. Em todos os casos foi obtido $x_{\text{ótimo}} = (1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 3; 3; 1)$, para simplificar foram omitidos os valores médios e desvios padrões de $x_{\text{ótimo}}$ na tabela. Deve-se ressaltar que, para estes resultados, todas as restrições foram obedecidas.

Além disso, pode-se observar na Tab. 4.6 que apenas a simulação com 32 processadores não apresentou valores super lineares para o speedup e a eficiência.

Problema Restrito 2: O segundo problema teste, dado pela Eq. (4.1.16), considera sete variáveis, cujos limites laterais são $-10 \leq x_i \leq 10, \ i = 1, \cdots, 7$ e quatro restrições representadas pelas Eqs. de (4.1.17) a (4.1.20).

$$\begin{aligned} min\ f(x) = \ & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + \\ & 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \end{aligned} \tag{4.1.16}$$

Tabela 4.7: Problema restrito 2.

Número de processadores	Tempo em segundos	Speedup	Eficiência	$f(x_{\text{ótimo}})$
1	70,72	1	1	$f = 680,6300573$ $\overline{f} = 680,6316054^{*}$ $\sigma = 0,0118828^{**}$
2	32,88	2,15	1,07	$f = 680,6300573$ $\overline{f} = 680,6313112^{*}$ $\sigma = 0,0090316^{**}$
4	16,99	4,16	1,04	$f = 680,6300573$ $\overline{f} = 680,6319933^{*}$ $\sigma = 0,0146679^{**}$
8	8,54	8,28	1,03	$f = 680,6300573$ $\overline{f} = 680,6326813^{*}$ $\sigma = 0,0199507^{**}$
16	3,98	17,77	1,11	$f = 680,6300573$ $\overline{f} = 680,6333892^{*}$ $\sigma = 0,0258041^{**}$
32	3,18	22,23	0,69	$f = 680,6300573$ $\overline{f} = 680,6340817^{*}$ $\sigma = 0,0330548^{**}$

* Valores médios e ** desvios padrões.

Sujeito a:

$$g_1(x) = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \leq 0 \tag{4.1.17}$$

$$g_2(x) = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \leq 0 \tag{4.1.18}$$

$$g_3(x) = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \leq 0 \tag{4.1.19}$$

$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \tag{4.1.20}$$

Os pontos ótimos calculados podem ser observados na Tab. 4.7. De forma similar ao exemplo anterior todas as restrições foram obedecidas e, independente do número de processadores utilizados, os valores ótimos encontrados para este problema foi $x_{\text{ótimo}}$=(2,330499; 1,951372; -0,4775414; 4,365726; -0,6244870; 1,038131; 1,594227).

Novamente, pode-se observar na Tab. 4.7, que apenas a simulação com 32 processadores não apresentou valores super lineares para o speedup e a eficiência.

Os excelentes resultados encontrados nos problemas resolvidos, restritos e irrestritos, utilizando Evolução Diferencial Melhorada em Paralelo permite validar o algoritmo, pois o mesmo mostrou-se preciso e robusto.

Os valores obtidos com as métricas speedup e eficiência indicam que toda a capacidade computacional disponível no sistema foi utilizada para ganhar velocidade na execução do algoritmo em paralelo visto que em todos os problemas resolvidos com 2, 4, 8 e 16 processadores obteve-se ou eficiência super linear ou quase linear, pois o valor da eficiência nestes casos ou foi maior que 1 ou muito próximo à 1. No entanto, ao usar 32 processadores para resolver os problemas apresentados a eficiência diminui ficando em torno de 70%, isto é resultante do aumento da comunicação entre os processadores.

A eficiência super-linear do código implementado em paralelo ocorre principalmente devido ao fato de que o algoritmo desenvolvido converge mais rapidamente quando divide a população em subpopulações. Além disso, também há independência entre os processos, independência esta que faz com que o tempo gasto entre as comunicações seja muito pequena.

Estes excelentes resultados obtidos com estes primeiros testes indicam que o algoritmo da EDMP está altamente paralelizável e que a comunicação entre os processadores é mínima. Portanto, pode-se garantir grande diminuição do tempo de execução do programa em relação ao algoritmo sequencial.

Nas próximas seções serão apresentadas simulações de problemas práticos da engenharia mecânica cujos resultados também confirmarão o quanto o algoritmo da EDMP é rápido nas suas execuções.

4.2 Resolução de Sistemas Lineares de dimensão elevada

A descrição e a solução de sistemas lineares são de grande importância em diversas áreas das ciências aplicadas. Segundo [31], mais de 75% dos problemas matemáticos encontrados em aplicações científicas e industriais envolvem a resolução de sistemas lineares em algum estágio. No entanto, estes sistemas geralmente possuem dimensão elevada o que dificulta a obtenção da sua solução de forma algébrica. De forma geral, os sistemas lineares podem ser resolvidos pelos métodos diretos e indiretos.

Os métodos diretos são aqueles que após um número finito e bem determinado de operações obtém-se a solução exata, a menos de erros de arredondamento, de um sistema linear. Para resolver um sistema linear por meio de um método direto é necessário modificar a matriz original dos coeficientes. Por isso, caso esta matriz seja esparsa estes métodos podem destruir sua esparsidade, no todo ou em parte.

Os métodos indiretos ou iterativos são aqueles que, a partir de uma aproximação inicial, obtém-se a solução aproximada do sistema linear

por meio de um processo iterativo. Esta solução aproximada depende de uma tolerância prefixada. Os métodos iterativos são classificados em estacionários e não-estacionários. Nos métodos estacionários cada solução aproximante é obtida a partir da solução anterior aplicando-se sempre o mesmo processo, ou seja, os operadores matriciais são constantes para todas as iterações. Nos métodos não-estacionários a matriz de iteração não é constante e por isso a cada iteração procura-se obter a melhor aproximação da solução de acordo com certas restrições e utilizando informações das iterações anteriores. Estes métodos se baseiam na teoria de otimização, procurando o mínimo de um funcional em uma direção de busca bem definida. A maioria destes métodos trata apenas de sistemas lineares onde a matriz dos coeficientes é simétrica e positiva definida. Existem algumas técnicas que podem ser utilizadas nos métodos iterativos para acelerar a convergência, por exemplo, o pré-condicionamento da matriz de coeficientes, que consiste em determinar uma matriz não singular, de forma que o novo sistema possua uma taxa de convergência maior ([44]).

Um sistema de equações lineares é uma coleção finita de variáveis e equações lineares (todas nas mesmas variáveis), consideradas em conjunto e normalmente apresentadas na forma matricial:

$$Ax^* = b, \tag{4.2.21}$$

onde A $\in \mathbb{R}^{n\times n}$ é a matriz dos coeficientes, b $\in \mathbb{R}^n$ é o vetor independente e x* é o vetor das incógnitas ou vetor solução, a ser determinado. Esta equação pode ser reestruturada como:

$$r(x) = b - Ax, \tag{4.2.22}$$

na qual o vetor solução x* foi substituído por um vetor genérico x e por isso há um vetor resíduo $r(x)$. É claro que, quando x = x* o resíduo será nulo, $r(x) = 0$.

O objetivo é aplicar técnicas heurísticas para encontrar a solução de sistemas lineares sem se preocupar com as características da matriz de coeficientes, mesmo que a mesma seja não simétrica e nem positiva definida. Além disso, não será necessária a utilização de pré-condicionadores para adequar o sistema aos métodos tradicionais. A metodologia utilizada consiste em minimizar o vetor resíduo Eq. (4.2.22) do sistema linear. Muitos métodos de otimização podem ser usados para resolver este problema de minimização. O algoritmo desenvolvido para a Evolução Diferencial Melhorada, implementado em paralelo, será aplicado na solução de grandes sistemas lineares.

4.2.1 Problema de Identificação de Forças Dinâmicas

Um processo de identificação de forças pode ser conduzido por meio de um modelo matemático que relacione as respostas medidas de um sistema mecânico com as possíveis forças aplicadas sobre o mesmo.

Os modelos de resposta são construídos a partir de equações diferencias que relacionam as entradas com as saídas do sistema em observação.

Os modelos de respostas podem ser estabelecidos por meio de modelagens nas quais os valores das saídas do sistema são obtidas por meio de experimentos com valores das entradas conhecidas. Assim, não é necessário ter conhecimento sobre a estrutura física interna do sistema.

O problema inverso da identificação de forças consiste em estimar as forças aplicadas sobre um sistema levando em consideração um modelo matemático e as respostas observadas. O modelo matemático inverso é determinado invertendo as equações matemáticas resultantes da modelagem do sistema físico.

Durante a modelagem de processos de identificação de forças é comum encontrar sistemas lineares mal condicionados que muitas vezes não podem ser resolvidos pelos métodos convencionais. Este fato justifica o uso de técnicas heurísticas para obter a solução de tais sistemas que possuem grande relevância na engenharia.

Serão considerados nesta seção dois problemas apresentados por [44]: um teórico e um experimental. No problema teórico o sistema é excitado por uma força harmônica. Em seguida um problema real de identificação indireta de forças é resolvido considerando os dados experimentais.

As simulações com EDMP, desta seção, foram executadas no cluster do Laboratório de Computação Científica Aplicada e Tecnologia de Informação - LCCATI da FACIP/UFU que possui as seguintes características:

- Frontend: 16 núcleos de processamento, 64GB de memória RAM, 1 adpatador Infiniband QDR X4 (40 Gbps) e duas unidades RAID (3TB home e 12TB tmp1) que são exportadas para todo o cluster;

- Oito (08) Nós computacionais, cada um com 64 núcleos de processamento, 128 GB de memória RAM, 2 adaptadores Infiniband QDR X4 (40 Gbps cada adaptador). Performance de pico de 8.8 GFlops por núcleo (4.5 TFLOPS total + 0.25 TFLOPS no frontend disponíveis para pequenas tarefas);

- Um nó de serviço com 8 núcleos, 64GB de memória RAM, 1 adpatador Infiniband QDR X4 (40 Gbps);

- Um servidor adicional com 16 núcleos de processamento, 64GB de memória RAM, 1 adpatador Infiniband QDR X4 (40 Gbps) e uma unidade de storage com 28 TB (Por enquanto este servidor está inativo);
- Switch infiniband 36 portas QDR X4 (40 Gbps por porta);
- Switch ethernet 1Gb 48 portas.

Sistema excitado por uma força harmônica
O objetivo é resolver o seguinte sistema linear:

$$X = HF \tag{4.2.23}$$

onde

$$X = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{Bmatrix}; \quad F = \begin{Bmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \end{Bmatrix} \quad e \quad H = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{p-1} & h_{p-2} & \cdots & h_0 \end{bmatrix} \tag{4.2.24}$$

sendo que,

$$h(i,j) = A_0 \left(e^{-\xi\omega t(j+1)}\right) sen\left(\sqrt{1-\xi^2}\omega t(j+1)\right) \tag{4.2.25}$$

com i, j= 1, 2, $\cdots$, N-1, e $j \leq i$.

Considerando $A_0 = e^{-3}$, $\xi = 0,08$, $\omega = 2\pi 500$, dt= 1/8192, $t \in [0,1]$ e N = comprimento(t). A força é dada por:

$$F = 5sen(2\pi 300t) + 1,5cos(2\pi 300t) \tag{4.2.26}$$

Visto que se conhece a matriz H e o vetor X, o sistema linear (4.2.23) será resolvido usando EDMP e o método iterativo Resíduo Mínimo Generalizado (GMRES) para encontrar o vetor F. A função objetivo é dada pelo resíduo $r = |X - HF|$. A modelagem completa deste problema de identificação de forças dinâmicas pode ser visto em [44]. Segundo [49], o GMRES é um método iterativo utilizado para resolver sistemas lineares que tem a propriedade de minimizar a cada iteração a norma do vetor residual sobre o subespaço de Krylov. O algoritmo é derivado a partir do processo de Arnoldi por construção da base orthogonal L_2 do subespaços de Krylov. O GMRES pode ser considerado como uma generalização do algoritmo MINRES de Paige e Saunders e é teoricamente equivalente ao método do Resíduo Conjugado Generalizado (GCR). Uma descrição mais detalhada do GMRES pode ser vista em [44].

A fim de avaliar o algoritmo EDMP foram realizados três testes com diferentes quantidades de indivíduos na população com os seguintes parâmetros: fator de diferença $F = 0{,}7$ e probabilidade de cruzamento $CR = 0{,}9$. O critério de parada do algoritmo EDMP adotado foi o número máximo de gerações da população e a verificação de sua estagnação. Nos três testes foram utilizados 80 processadores do cluster. Em seguida, as soluções numéricas encontradas com EDMP serão comparadas com as soluções analíticas e as obtidas pelo GMRES no MATLAB®.

Teste 1

No primeiro teste foram adotados $t \in [0; 0,01]$, 1600 indivíduos na população inicial o que equivale à 20 indivíduos por processador e, no máximo, 100 iterações para o critério de parada. Neste caso a matriz H tem 82×82 entradas. Foram executadas 114.560 avaliações da função objetivo em 6,95 minutos. A Fig. 5.11 apresenta a comparação entre as soluções obtidas e o erro relativo entre a solução analítica e a encontrada com EDMP. O valor do resíduo na solução ótima encontrada por EDMP foi $r = |X - HF| = 9,91 \times 10^{-5}$.

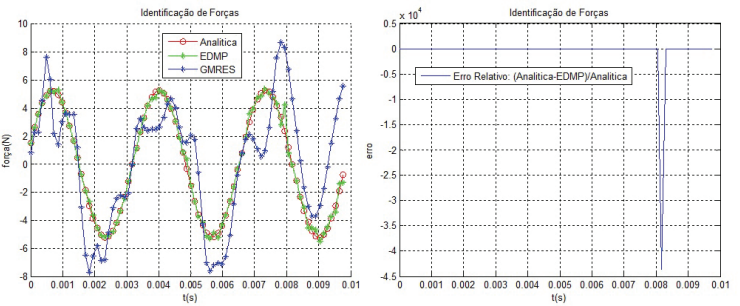


Figura 4.2: Solução analítica e numérica encontrada com EDMP e GMRES - Teste 1.

Teste 2

A fim de tentar melhorar a solução obtida no teste 1, uma nova simulação foi executada considerando 4800 indivíduos na população inicial, o que equivale à 60 indivíduos por processador e, no máximo, 200 iterações para o critério de parada. Foram executadas 285.660 avaliações da função objetivo em 43,84 minutos. A Fig. 5.12 apresenta a comparação entre as soluções obtidas e o erro relativo entre a solução analítica e a encontrada com EDMP. O valor do resíduo na solução ótima encon-

trada por EDMP foi $r = |X - HF| = 9,60 \times 10^{-5}$.

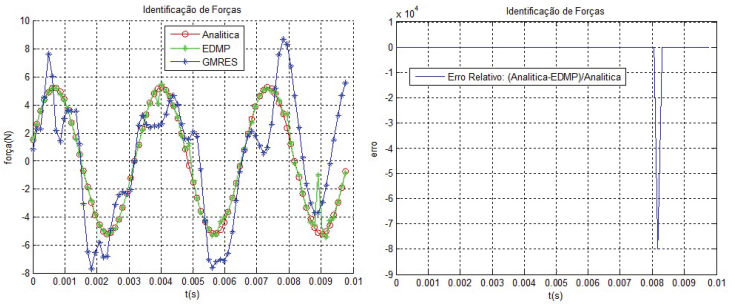


Figura 4.3: Solução analítica e numérica encontrada com EDMP e GMRES - Teste 2.

Analisando os resultados obtidos com os testes 1 e 2 pode-se observar que as respostas dos valores da força F encontradas ao longo do tempo são muito satisfatórias já que estão muito próximas às respostas obtidas por meio da solução analítica. O erro relativo entre as soluções analítica e EDMP também mostrou-se muito pequeno, salvo em alguns pequenos intervalos onde a solução aproximada por EDMP afastou-se da solução analítica.

Há poucas diferenças entre as soluções encontradas com os testes 1 e 2. O valor do resíduo r é praticamente o mesmo, no entanto, no teste 2, apesar de haver três vezes mais indivíduos e 100 iterações a mais que o teste 1, o valor do resíduo é sutilmente maior, isto é, a diferença em módulo entre os valores dos resíduos é inferior à 0,0000032.

Portanto, pelo menos para este problema, é inviável o uso de uma população muito grande e de muitas iterações, pois tais escolhas sobrecarregam desnecessariamente os processadores e aumentam significativamente o tempo de execução do algoritmo.

Teste 3

Neste teste o objetivo é comparar as soluções entre EDMP e GMRES em simulações onde o sistema linear é "grande". Para tanto, foi utilizado $t \in [0; 0,03]$, 3200 indivíduos da população inicial, o que equivale à 40 indivíduos por processador e, no máximo, 100 iterações para o critério de parada. A matriz H tem 246×246 entradas e foram executadas 557.680 de avaliações da função objetivo em 1,7 horas. A Fig. 5.13 apresenta a comparação entre as soluções obtidas e o erro relativo entre a solução analítica e a encontrada com EDMP. O valor do resíduo na solução ótima encontrada por EDMP foi $r = |X - HF| = 9,82 \times 10^{-5}$.

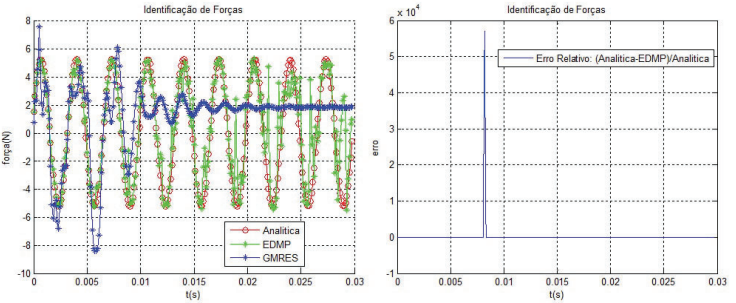


Figura 4.4: Solução analítica e numérica encontrada com EDMP e GMRES - Teste 3.

Este teste exemplifica uma das dificuldades do GMRES que é obter a solução de "grandes" sistemas lineares. A partir de determinado momento o método começa a divergir. A solução obtida por EDMP no teste 3 é muito mais precisa que a do GMRES e o valor do resíduo encontrado mostrou-se baixo. Uma desvantagem é o tempo de execução do algoritmo. Para esta simulação o GMRES levou 0,7 segundos para encontrar a solução do sistema porém, tal solução é bastante diferente da solução analítica. Embora o esforço computacional do EDMP seja grande, acarretando em muito tempo de execução, a solução por ele obtida está próxima da solução analítica.

Na área científica há várias pesquisas onde os sistemas lineares são representados por milhares ou até mesmo milhões de variáveis. No trabalho de [21], por exemplo, é resolvido um sistema linear da ordem de 55.296 que precisou de mais de 4 dias para obter a solução por meio do método Gradiente Conjugado implementado em paralelo.

Os testes realizados com EDMP permitem concluir que o método proposto se apresenta como uma alternativa para resolução de sistemas lineares de grande porte.

Na próxima subseção será resolvido um sistema linear com 3.500 variáveis para avaliar o desempenho do algoritmo EDMP na obtenção da solução ótima.

4.2.2 Sistema Dinâmico Usando Dados Experimentais

Nesta simulação o sistema linear deriva de um problema de identificação indireta de forças. O problema consiste em determinar a força aplicada a um sistema formado por três placas conectadas por três conjuntos de quatro lâminas de aço dispostas em paralelo conforme apresentado por

[44]. No domínio de baixas frequências o sistema pode ser modelado como um sistema de três graus de liberdade pouco amortecido. A Fig. 5.14 ilustra o sistema mecânico utilizado nos experimentos.

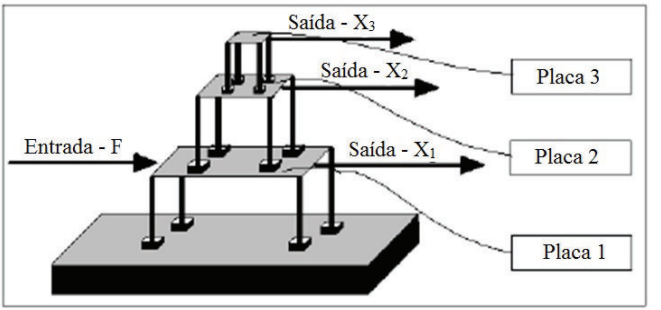


Figura 4.5: Ilustração do sistema mecânico utilizado nos experimentos. Fonte: [44] pg. 119.

Neste experimento uma força de excitação por impacto F é aplicada na coordenada 1 (Placa 1). As correspondentes respostas temporais (X_1, X_2 e X_3) foram medidas nas coordenadas 1, 2 e 3. Tanto a força como as respostas temporais foram medidas somente na direção horizontal. Mais detalhes do experimento pode ser visto em [44].

Seja a integral de convolução de Duhamel dada pela Eq. (4.2.27):

$$x(t) = \int_{-\infty}^{\infty} [h(t-\tau)]\{f(\tau)\}d\tau \tag{4.2.27}$$

onde $h(t)$ é a F.R.I., $f(t)$ é a força excitadora em função do tempo t e $x(t)$ é a amplitude da vibração em resposta no tempo t.

Como no experimento há três locais de medição e uma força de excitação, a Eq. (4.2.27) pode ser desenvolvida da seguinte forma:

$$\begin{aligned} x_1(t) &= \textstyle\int_0^t [h_{11}(t-\tau)]\{f(t)\}d\tau \\ x_2(t) &= \textstyle\int_0^t [h_{21}(t-\tau)]\{f(t)\}d\tau \\ x_3(t) &= \textstyle\int_0^t [h_{31}(t-\tau)]\{f(t)\}d\tau \end{aligned} \tag{4.2.28}$$

sendo que a F.R.I. $h_{i1}(t)$ representa as respostas na coordenada i devido à excitação aplicada na placa 1.

A discretização Δt do tempo de amostragem da Eq. (4.2.28) é dada por:

$$\begin{aligned} x_1(k\Delta t) &= \textstyle\sum_{i=0}^{k} h_{11}[(k-i)]f(i)\Delta t \\ x_2(k\Delta t) &= \textstyle\sum_{i=0}^{k} h_{21}[(k-i)]f(i)\Delta t \\ x_3(k\Delta t) &= \textstyle\sum_{i=0}^{k} h_{31}[(k-i)]f(i)\Delta t \end{aligned} \tag{4.2.29}$$

com $k = 0, 1, 2, \cdots, p-1$.

As Eqs. (4.2.29) serão agrupadas em um sistema de $3p$ equações e p incógnitas, da seguinte forma matricial:

$$\{X\} = [h]\{F\} \tag{4.2.30}$$

onde

$$\{X\} = \left\{ \begin{array}{c} \{X(0)\} \\ \{X(1)\} \\ \vdots \\ \{X(p-1)\} \end{array} \right\}, \tag{4.2.31}$$

$$[h] = \left[\begin{array}{cccc} [h(0)] & [0] & \cdots & [0] \\ [h(1)] & [h(0)] & \cdots & [0] \\ \vdots & \vdots & \ddots & \vdots \\ [h(p-1)] & [h(p-2)] & \cdots & [h(0)] \end{array} \right] \tag{4.2.32}$$

e

$$\{F\} = \left\{ \begin{array}{c} F(0) \\ F(1) \\ \vdots \\ F(p-1) \end{array} \right\} \tag{4.2.33}$$

Sendo que,

$$\{X(i)\} = \left\{ \begin{array}{c} x_1(i) \\ x_2(i) \\ x_3(i) \end{array} \right\}, \quad [h(i)] = \Delta t \left[\begin{array}{c} h_{11}(i) \\ h_{21}(i) \\ h_{31}(i) \end{array} \right] \quad e \quad [0] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array} \right] \tag{4.2.34}$$

com $i = 0, 1, \cdots, p-1$.

Para calcular o valor do vetor $\{F\}$ a partir da Eq. (4.2.30) será utilizado o método dos mínimos quadrados. Assim, tem-se que:

$$\{F\} = ([h]^T [h])^{-1} [h]^T \{X\} \tag{4.2.35}$$

Conforme apresentado por [44], os sinais de força e de acelerações, sinais de entrada e saída respectivamente, foram amplificados usando amplificadores de sinais e enviados para a placa de aquisição e posteriormente armazenados em um microcomputador. Cada sinal recebido foi observado no intervalo de 0 a 13,67s discretizado em 3500 pontos igualmente espaçados ($\Delta t = 3,9 \times 10^{-3}$s). A solução do sistema dado pela Eq. (4.2.30) será obtida para as F.R.I.s $h_{11}(t)$, $h_{21}(t)$ e $h_{31}(t)$ por meio da minimização do resíduo $r = |\{X\} - [h]\{F\}|$ pelo método EDMP. Visto que a força exata é conhecida, esta será utilizada para comparar com a força identificada a fim de avaliar a precisão do procedimento. Em seguida, serão apresentados nas Figs. 5.15 à 5.22 uma

comparação entre os resultados encontrados e os valores exatos da força de excitação e da aceleração para cada uma das placas considerando os erros cometidos.

Na simulação realizada adotou-se 160 indivíduos na população inicial divididos entre 16 processadores do cluster assim, cada processador executou o algoritmo com 10 indivíduos. Além disso, foi adotado os seguintes parâmetros para EDMP: fator de diferença $F = 0{,}8$ e probabilidade de cruzamento $CR = 0{,}5$. O critério de parada adotado foi o número máximo de gerações da população e a verificação de sua estagnação.

O tempo de execução do EDMP foi 3,86 horas para um total de 22.000 avaliações da função objetivo. O valor do resíduo no ponto ótimo é $r = 0,0594$.

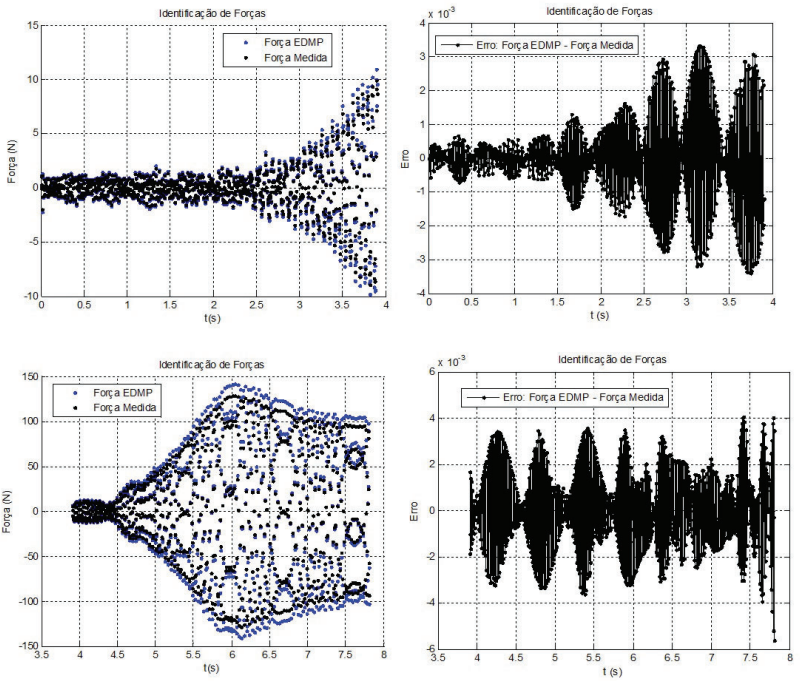


Figura 4.6: Comparativo entre a Força aproximada por EDMP e os valores medidos para t=0s a t=8s.

Os resultados obtidos com EDMP foram muito satisfatórios já que o erro cometido é pequeno e aceitável. A maior vantagem de se usar EDMP é o fato de não se preocupar com condicionamento de matrizes

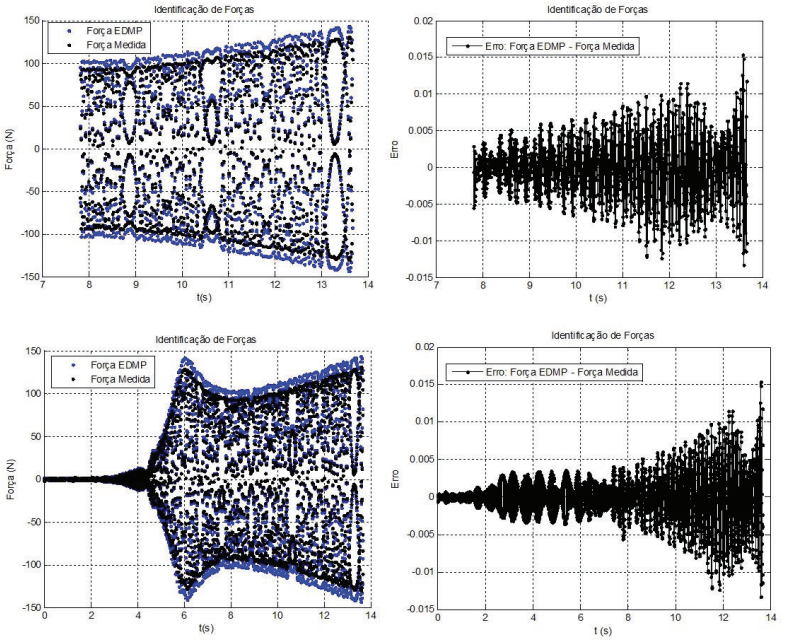


Figura 4.7: Comparativo entre a Força aproximada por EDMP e os valores medidos para t=0s a t=14s.

e o da matriz não precisar ser quadrada. Além disso, o problema de identificação de forças dinâmicas foi resolvido sem usar nenhuma técnica de regularização da matriz de coeficientes, visto que esta é uma matriz retangular formada por três blocos triangulares.

Uma desvantagem do EDMP é o tempo de execução já que a solução de grandes sistemas lineares requer grandes recursos computacionais uma vez que o EDMP trabalha com avaliação da função objetivo. Além disso, visto que há um grande volume de dados para armazenar, a solução destes problemas precisa de um cluster com memória distribuída ou supercomputadores com grande memória compartilhada.

Visto que a solução de sistemas lineares desempenha um papel importante nas engenharias há muitos métodos numéricos para resolução sistemas lineares, mas muitas também são as suas limitações. Em [48] pode-se encontrar um estudo das seguintes metodologias de solução de sistemas lineares: Fatoração LU, Fatoração Cholesky, Jacobi, Gauss-Seidel. Estes métodos são mais eficazes para matrizes quadradas. No trabalho de [54] é feita uma análise dos métodos Gradiente Conjugado, GMRES e QR para Autovalores. Tais métodos, geralmente, são mais

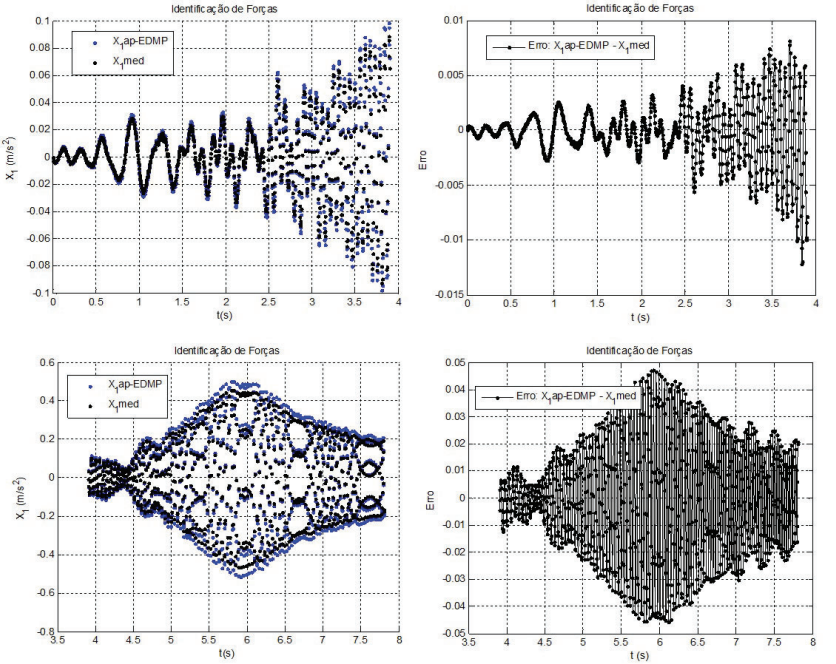


Figura 4.8: Comparativo entre a resposta X_1 aproximada por EDMP e os valores medidos para t=0s a t=8s.

eficientes para matrizes simétricas positiva definida.

Embora a solução de sistemas lineares seja conceitualmente simples, na prática podem surgir muitos desafios. Acredita-se, devido as simulações realizadas nesta pesquisa, que o EDMP seja uma alternativa promissora, principalmente devido a praticidade de não se preocupar com as características da matriz dos coeficientes do sistema linear.

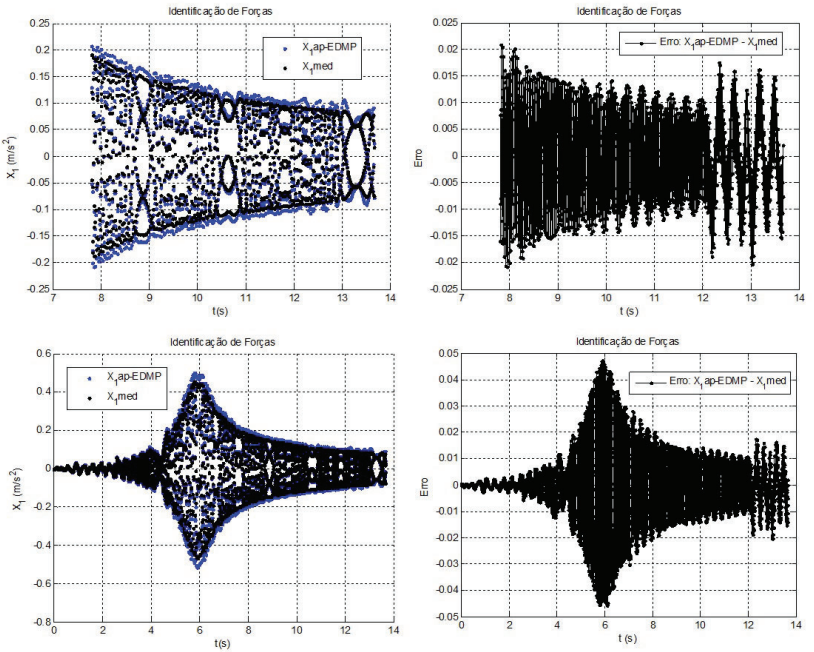


Figura 4.9: Comparativo entre a resposta X_1 aproximada por EDMP e os valores medidos para t=0s a t=14s.

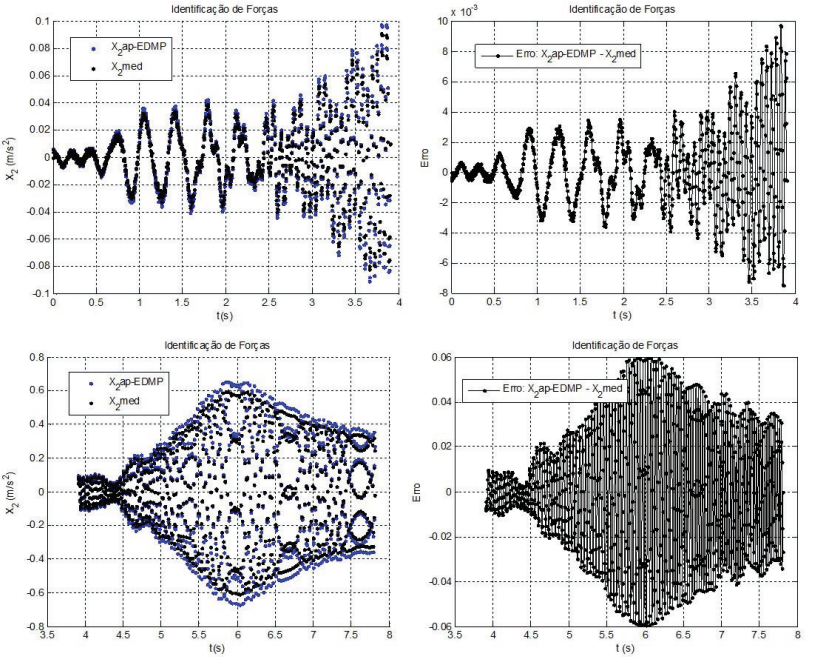


Figura 4.10: Comparativo entre a resposta X_2 aproximada por EDMP e os valores medidos para t=0s a t=8s.

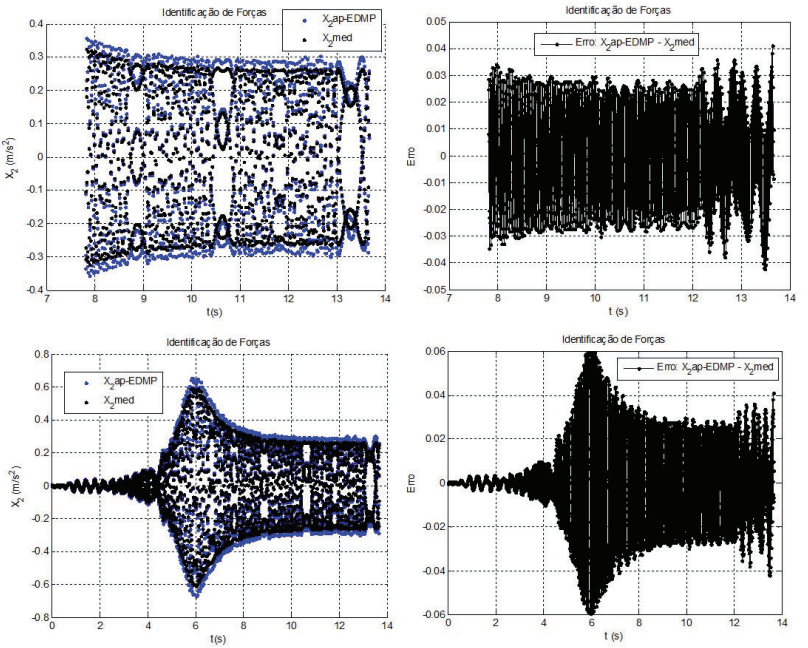


Figura 4.11: Comparativo entre a resposta X_2 aproximada por EDMP e os valores medidos para t=0s a t=14s.

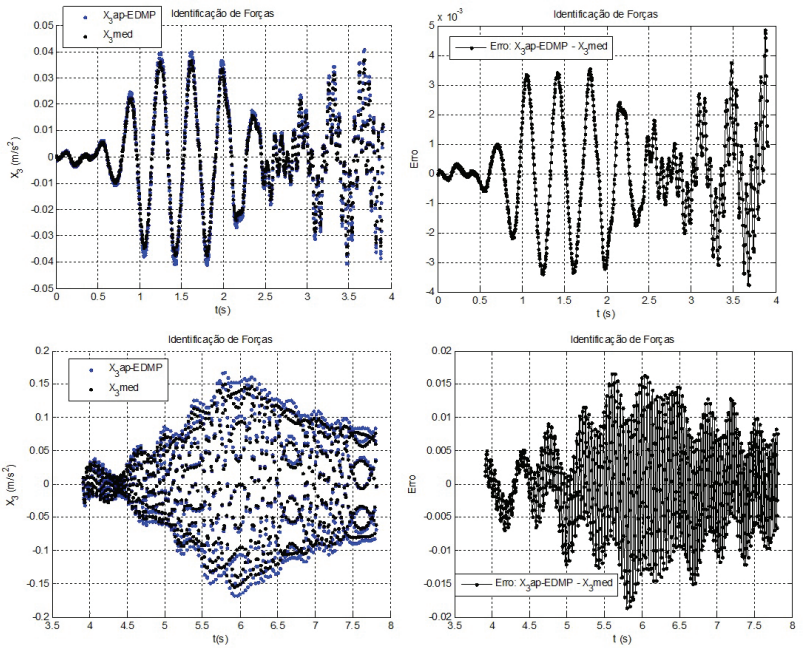


Figura 4.12: Comparativo entre a resposta X_3 aproximada por EDMP e os valores medidos para t=0s a t=8s.

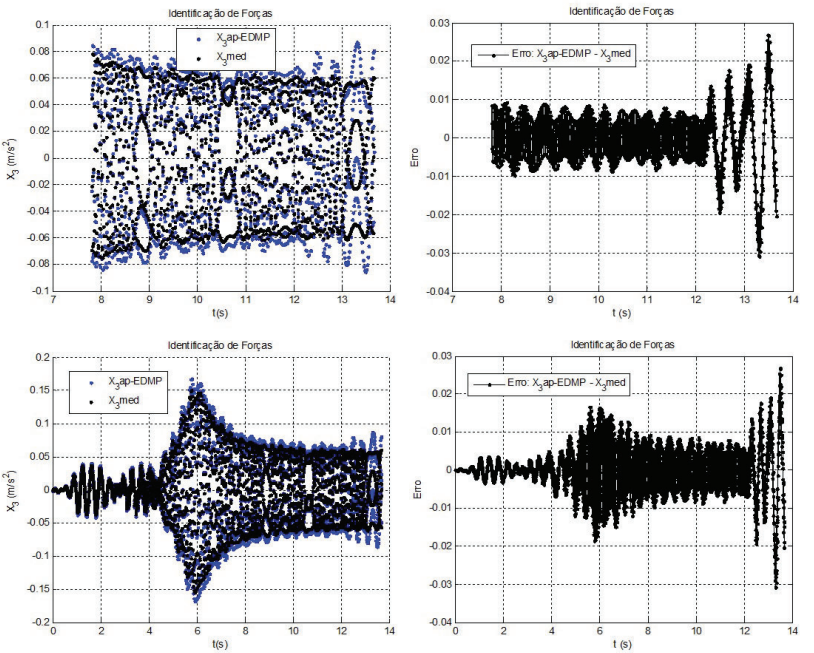


Figura 4.13: Comparativo entre a resposta X_3 aproximada por EDMP e os valores medidos para t=0s a t=14s.

4.3 Otimização de Sistemas Robóticos

O objetivo desta seção é avaliar o algoritmo Evolução Diferencial Melhorada implementado em paralelo na resolução de um problema prático da engenharia mecânica que consiste em obter o projeto ideal de um robô manipulador levando em conta as características do seu espaço de trabalho, conforme apresentado por [36] e [51]. Para esta finalidade, um problema multiobjetivo de otimização é formulado para se obter os parâmetros ideais para o robô. Os resultados mostram que o procedimento representa uma alternativa promissora para este tipo de problema.

Será considerado os robôs manipuladores com três juntas rotacionais, cuja abreviação é dada por 3R, conforme Fig. 4.14.

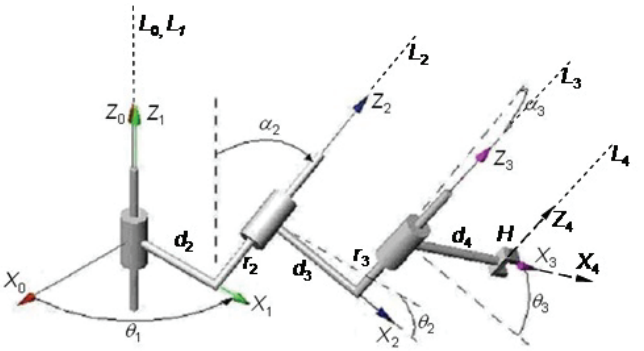


Figura 4.14: Esquema cinemático de um robô manipulador 3R com seus parâmetros de projeto. Fonte: [51].

O mecanismo de um manipulador em série é um conjunto cinemático constituído de uma sucessão de corpos rígidos ligados entre si por juntas rotacionais e/ou prismáticas. Uma forma de representação cinemática comum é a utilização dos parâmetros de Denavit-Hartenberg (DH). Dois sistemas de referência R_{j-1} e R_j são relacionados pela matriz de transformação T_j^{j-1}, dada por:

$$T_j^{j-1} = \begin{pmatrix} c_j & -s_j & 0 & d_j \\ c\alpha_j s_j & c\alpha_j c_j & -s\alpha_j & -r_j s\alpha_j \\ s\alpha_j s_j & s\alpha_j c_j & c\alpha_j & r_j c\alpha_j \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

sendo $c_j = cos(\theta_j), s_j = sen(\theta_j), c\alpha_j = cos(\alpha_j)$ e $s\alpha_j = sen(\alpha_j), j = 1, ..., n+1$.

Assim, ao efetuador de um robô de n graus de liberdade é associado um sistema de coordenadas, T_n^0, que é uma matriz de transformação homogênea $T_n^0 = T_1^0 \cdot T_2^1 \cdots T_j^{j-1} T_{j+1}^j \cdots T_n^{n-1}$.

Trabalhando com estas matrizes, obtém-se o Modelo Geométrico Direto (MGD) de um manipulador 3R, dado por:

$$\begin{aligned} x = \; & [d_2 + (r_3 s\alpha_3 - d_4 c\alpha_3 s_3)\, s_2 + (d_3 + d_4 c_3)\, c_2]\, c_1 \\ & + \{s\alpha_2\, (r_2 + r_3 c\alpha_3 + d_4 s\alpha_3 s_3) \\ & + c\alpha_2\, [(r_3 s\alpha_3 - d_4 c\alpha_3 s_3)\, c_2 - (d_3 + d_4 c_3)\, s_2]\}\, s_1 \\ y = \; & [d_2 + (r_3 s\alpha_3 - d_4 c\alpha_3 s_3)\, s_2 + (d_3 + d_4 c_3)\, c_2]\, s_1 \\ & - \{s\alpha_2\, (r_2 + r_3 c\alpha_3 + d_4 s\alpha_3 s_3) \\ & + c\alpha_2\, [(r_3 s\alpha_3 - d_4 c\alpha_3 s_3)\, c_2 - (d_3 + d_4 c_3)\, s_2]\}\, c_1 \\ z = \; & c\alpha_2\, (r_2 + r_3 c\alpha_3 + d_4 s\alpha_3 s_3) \\ & - s\alpha_2\, [(r_3 s\alpha_3 - d_4 c\alpha_3 s_3)\, c_2 - (d_3 + d_4 c_3)\, s_2] \end{aligned} \tag{4.3.36}$$

Com as Eqs. (4.3.36) pode-se obter o determinante da matriz Jacobiana J. Será utilizada nesta pesquisa a matriz Jacobiana de Base obtida por [36] e [6] que é dada por:

$$J = \begin{pmatrix} -s_3 c_2 d_4 - c_2 r_2 & r_3 & -s_3 d_4 \\ s_3 s_2 d_4 + s_2 r_2 & d_3 + c_3 d_4 & 0 \\ c_2 d_3 + c_2 c_3 d_4 + d_2 + s_2 r_3 & 0 & c_3 d_4 \end{pmatrix} \tag{4.3.37}$$

cujo determinante é dado por:

$$det(J) = d_4 \left\{(d_3 + d_4 c_3)[d_2 s_3 + (d_3 s_3 - r_2 c_3) c_2] + r_3 s_2 (d_3 s_3 - r_2 c_3)\right\}. \tag{4.3.38}$$

Um problema de otimização multiobjetivo será formulado para obtenção dos parâmetros ideais do robô manipulador. A formulação matemática para calcular o volume do espaço de trabalho, a rigidez do sistema e a destreza do robô será dada em seguida.

Formulação do Problema de Otimização

O trabalho proposto consiste na síntese dimensional de um manipulador 3R ortogonal (α_2= -90º e α_3 = 90º), onde o projeto ótimo corresponde àquele que otimiza algumas características desejáveis para a melhor performance do robô à executar tarefas. O objetivo é a obtenção de um projeto ótimo que considera a maximização do volume de trabalho do manipulador (V), a maximização da rigidez do mecanismo (R) e a otimização de sua destreza por meio da minimização do índice de isotropia (D). Assim, o problema de otimização pode ser formulado como:

$$\text{Maximizar } F(X) = [V \; -D \; R] \tag{4.3.39}$$

onde

$$X = (1, d_3, d_4, r_2, r_3)^T, \qquad 0,1 \leq d_3, d_4, r_2, r_3 \leq 3,0 \tag{4.3.40}$$

sendo que as variáveis d_3, d_4, r_2 e r_3 estão representadas na Fig. 4.14.

O trabalho atual utiliza a modelagem desenvolvida por [37] e [39], que aplicou em sua pesquisa Algoritmo Genético e Evolução Diferencial para solucionar o problema de otimização.

Volume do Espaço de Trabalho

O volume do espaço de trabalho, V, é o volume do sólido de revolução obtido pela rotação da seção radial em torno do eixo z. Assim, usando o Teorema de Pappus-Guldin, conforme apresentado na Fig. 4.15, o volume é dado através da equação:

$$V = 2\pi r_g A, \tag{4.3.41}$$

sendo A a área da seção radial plana que é coberta pela família de curvas e r_g a coordenada do baricentro.

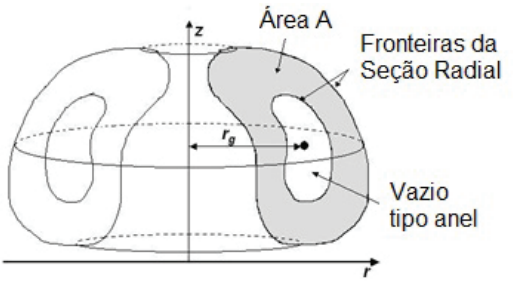


Figura 4.15: Cálculo do volume do espaço de trabalho de manipuladores 3R. Fonte: [36].

Esta pesquisa propõe uma formulação numérica para aproximar o cálculo da área da seção radial através de sua discretização em uma malha retangular ([51]). Inicialmente, devem-se obter os valores extremos dos vetores r e z, ou seja,

$$\begin{array}{lll} r_{min} = min\{r\} & e & r_{max} = max\{r\} \\ z_{min} = min\{z\} & e & z_{max} = max\{z\} \end{array} \tag{4.3.42}$$

Adotando-se o número de subintervalos desejados para a discretização ao longo de r e z (n_r e n_z), pode-se calcular as dimensões das áreas elementares da malha através das seguintes expressões:

$$\Delta_r = \frac{r_{max} - r_{min}}{n_r} \quad e \quad \Delta_z = \frac{z_{max} - z_{min}}{n_z} \tag{4.3.43}$$

Utilizando as Eqs. (4.3.36), do modelo geométrico direto do manipulador, calcula-se todos os pontos da família de curvas que compõem a seção radial do espaço de trabalho. Dado um determinado ponto (r, z), determina-se sua posição dentro da malha de discretização, através do seguinte controle de índices:

$$i = int\left(\frac{r - r_{min}}{\Delta_r}\right) + 1 \quad e \quad j = int\left(\frac{z - z_{min}}{\Delta_z}\right) + 1 \tag{4.3.44}$$

Conforme mostrado no esquema da Fig. 4.16, o ponto da malha que pertence ao espaço de trabalho será identificado como $P_{ij} = 1$, caso contrário terá valor nulo, ou seja:

$$P_{ij} = \begin{cases} 0, & se\ P_{ij} \notin W(H) \\ 1, & se\ P_{ij} \in W(H) \end{cases} \tag{4.3.45}$$

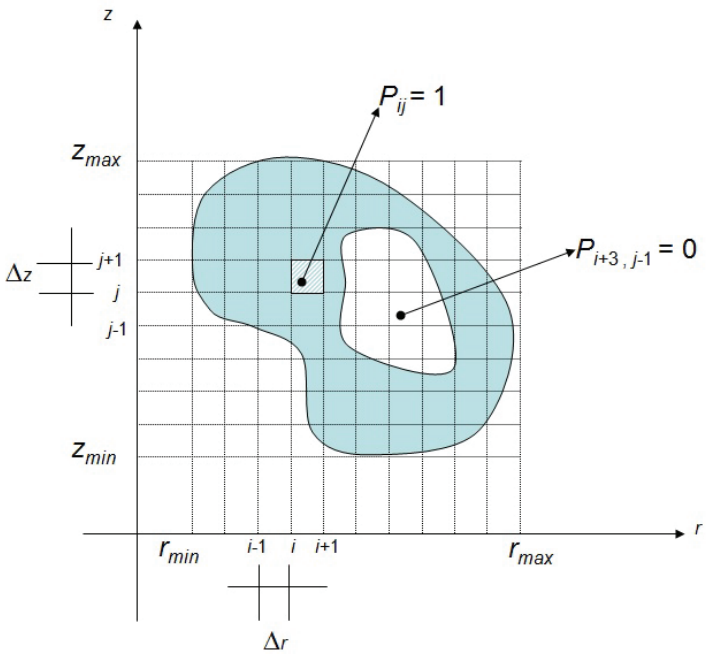


Figura 4.16: Discretização da seção radial usando malha retangular. Fonte [38].

Desta forma, a área total é obtida pela soma de todas as áreas elementares da malha que estão contidas, totalmente ou parcialmente, na

seção radial, conforme Eq. (4.3.46). Observa-se que apenas os pontos pertencentes ao espaço de trabalho contribuem para o cálculo da área:

$$A = \sum_{i=1}^{i_{max}} \sum_{j=1}^{j_{max}} (P_{ij} \Delta_r \Delta_z) \tag{4.3.46}$$

A coordenada do baricentro é calculada considerando a soma dos baricentros de cada área elementar, dividida pela área total, dada por:

$$r_g = \frac{\sum_{i=1}^{i_{max}} \sum_{j=1}^{j_{max}} (P_{ij} \Delta_r \Delta_z) \left((i-1)\, \Delta_r + \frac{\Delta_r}{2} + r_{min} \right)}{A} \tag{4.3.47}$$

Finalmente, conhecendo-se os valores da área e do baricentro da seção radial, dados pelas Eqs. (4.3.46) e (4.3.47), pode-se calcular o volume do espaço de trabalho do manipulador usando a Eq. (4.3.41). Desta forma, a Eq. (4.3.41) representa a função objetivo a ser maximizada para obter o maior volume do espaço de trabalho possível.

Rigidez do Mecanismo Serial

Pode-se definir rigidez como sendo a capacidade de um sistema mecânico de suportar cargas sem grandes mudanças em sua geometria. Assim, a rigidez é uma característica mecânica que descreve o comportamento de uma estrutura sujeita às forças estáticas em termos da deflexão elástica. Desta forma, o estudo da rigidez de um sistema equivale a obter a matriz de rigidez, K, da estrutura analisada, que representa a medida da capacidade da estrutura de resistir às deformações devido à ação de esforços externos.

A fim de se obter o modelo da rigidez da estrutura pode-se adotar a matriz de análise do mecanismo que lida com a estrutura como uma combinação de elementos e nós conforme proposto por [20] ou ainda com métodos baseados no cálculo da matriz Jacobiana do mecanismo serial de acordo com [22] e [19].

Neste trabalho, o modelo de rigidez da estrutura é obtido a partir da matriz Jacobiana do mecanismo serial, considerando os elementos como sendo molas. A matriz de rigidez do sistema de juntas do mecanismo no espaço cartesiano é dada por:

$$K = [J]^T K_j [J] \tag{4.3.48}$$

onde K_j é a matriz diagonal $n \times n$. Para o caso estudado, mecanismo serial com 3 juntas rotacionais, tem-se $K_j = diag[k_1, k_2, k_3]$ e J a matriz Jacobiana, dada na Eq. (4.3.37).

Além disso, os elementos da diagonal da matriz de rigidez são utilizados como os valores da rigidez do sistema. Estes elementos representam

a rigidez pura em cada direção, e refletem a rigidez/inflexibilidade das ferramentas da máquina de forma mais clara e direta. Assim, a função objetivo para otimizar a rigidez do sistema pode ser escrita conforme a Eq. (4.3.49). Neste caso, a rigidez R deve ser maximizada:

$$R = K_{11} + K_{22} + K_{33} \tag{4.3.49}$$

onde $K_{ii}, i = 1, 2, 3$, representa os elementos da diagonal da matriz de rigidez do mecanismo.

Destreza

O índice de desempenho de um sistema mecânico robótico é uma quantidade escalar que mede o quão satisfatório o sistema se comporta com relação à transmissão de movimento e força. Este índice pode ser definido para todos os tipos de sistemas mecânicos robóticos, particularmente, os manipuladores seriais. Existem vários índices de desempenho definidos na literatura que podem ser vistos em [43], [58], [59] e [60].

Neste trabalho, será adotado o índice de isotropia, D, que é definido como o índice de condicionamento da matriz Jacobiana J. Tal índice pode ser escrito como a relação entre o maior e o menor valor singular de J:

$$D = \frac{|\lambda_{max}(J)|}{|\lambda_{min}(J)|} \tag{4.3.50}$$

sendo que λ_{max} e λ_{min} significam, respectivamente, os valores singulares máximo e mínimo de J.

Na robótica, o índice de isotropia reflete a precisão resultante das velocidades operacionais de entrada a partir das velocidades articulares calculadas usando a inversa da matriz J. Este índice pode alcançar valores no intervalo $[1; \infty]$.

O valor 1 significa uma isotropia: o elipsoide das velocidades toma a forma de uma esfera. Fisicamente, quando o manipulador está em uma posição isotrópica, o efetuador terá a mesma facilidade para se deslocar em todas as direções. Assim, manipuladores isotrópicos são aqueles cuja Jacobiana pode alcançar valores isotrópicos, neste caso, $D = 1$. Quanto mais próximo de 1, maior a agilidade para executar tarefas no espaço de trabalho.

O índice de isotropia pode indicar distorção no espaço das variáveis. Quanto maior for esta distorção, maior será o índice de isotropia. Portanto, para a otimização da destreza, o índice de isotropia deve ser minimizado.

4.3.1 Simulação Numérica do Problema Irrestrito

Como mencionado anteriormente, o objetivo do esquema proposto para o projeto do manipulador é a otimização da dimensão do robô ortogonal

3R considerando o espaço de trabalho (V), a rigidez do mecanismo (R) e a destreza do manipulador (D), conforme mostrado nas Eqs. (4.3.39) e (4.3.40). Trata-se de um problema de otimização multiobjetivo e nesta pesquisa adotaremos o Método da Ponderação dos Objetivos e o Método do Critério Global, que são abordagens utilizadas na resolução de problemas nos quais há a presença de mais de um objetivo a ser otimizado ([17], [14], [18], [2]).

O volume do espaço de trabalho é dado pela Eq. (4.3.41) e a rigidez do mecanismo pela Eq. (4.3.49). A fim de otimizar a destreza, o índice de isotropia, dado pela Eq. (4.3.50), foi minimizado.

No intuito de avaliar a performance entre as técnicas ECE, ED e EDMP os seguintes pontos devem ser salientados:

- parâmetros da ED: tamanho da população (9), taxa de pertubação (0,8), probabilidade de cruzamento (0,6);
- parâmetros da EDMP: tamanho da população 36 indivíduos divididos entre 4 processadores, taxa de pertubação (0,8), probabilidade de cruzamento (0,6);
- parâmetros da ECE: tamanho da população (9 indivíduos em cada complexo), o ponto inicial é o ponto médio dos limites laterais das variáveis de projeto, a saber $x_0 = [1,45 \;\; 1,45 \;\; 1,45 \;\; 1,45]$;
- critério de parada usado em ED foi número máximo de iterações ou estagnação do valor da função objetivo. O critério de parada usado em ECE foi a série geométrica normalizada dos parâmetros menor do que 0,0001. O critério de parada usado em EDMP foi o número máximo de iterações;
- ED e EDMP foi executado 20 vezes para obter a média dos valores apresentados nas tabelas;
- parâmetros do robô considerados: $d_2 = 1$, $d_3 = x(1)$, $d_4 = x(2)$, $r_2 = x(3)$, $r_3 = x(4)$, $a_2 = -\pi/2$, $a_3 = \pi/2$, o tamanho do passo para calcular a Jacobiana: 0,03 e malha: 50×50;
- todas as simulações foram resolvidas usando um processador Intel® Core^{TM} i5-430M.

Nas seguintes Tabs., 4.8 e 4.9, estão apresentados os valores ótimos encontrados usando os métodos de otimização Evolução Diferencial, Evolução com Conjuntos Embaralhados (com 2, 4 e 8 conjuntos) e Evolução Diferencial Melhorada implementado em paralelo. É importante mencionar que parte dos resultados expostos nas Tabs. 4.8 e 4.9 foram apresentados por [11] no World Congress on Computational Mechanics.

Vale salientar que os resultados ótimos apresentados na Tab. 4.9 são dependentes dos valores dos coeficientes de ponderação. Esta tabela mostra o caso em que a destreza é priorizada (w_2=0,9) e o caso em que todas as prioridades são iguais ($w_1 = w_2 = w_3$=1/3). Aqui usamos o Método da Ponderação dos Objetivos na abordagem "a priori", ou seja, os pesos são definidos previamente e incorporados diretamente no processo de otimização visando priorizar um determinado critério.

Considerando que os valores ideais do volume do espaço de trabalho, da destreza e da rigidez são, V_{ideal}=2382.7412 [u.v.], D_{ideal}=1.0256 e R_{ideal}=125.0769 [u.r.], respectivamente, pode-se concluir que os métodos são eficazes em resolver os problemas visto que os resultados obtidos aproximam dos valores ideais. Comparando os resultados encontrados com ECE, ED e EDMP é possível observar que os valores são muito similares. No entanto, para o problema estudado, o método EDMP mostrou-se muito mais rápido. Por exemplo, para encontrar a solução ótima do problema proposto considerando a métrica L_{2R} (Tab. 4.8) o menor tempo de execução entre os algoritmos sequenciais foi de 3,05 horas enquanto EDMP precisou de apenas 45 segundos. Este comportamento também se repetiu considerando o método dos objetivos ponderados. Outros problemas de otimização multiobjetivo de sistemas robóticos irrestritos foram simulados com EDMP. Por exemplo, em [13] é apresentado e comparado as soluções obtidas com ED e EDMP para encontrar a solução ótima de sistemas robóticos considerando como variáveis de projeto os seguintes parâmetros geométricos do robô: $d_3, d_4, r_2, r_3, \theta_2$ e θ_3. Nesta simulação também foi usado o Método da Ponderação dos Objetivos e o Método do Critério Global para resolver o problema multiobjetivo dado pela Eq. (4.3.39). Novamente, os resultados obtidos com a ED e EDMP foram muito semelhantes. No entanto, enquanto ED precisou em torno de 11 a 30 horas, dependendo do método, para encontrar a solução, EDMP gastou em média 14 minutos.

As bem sucedidas aplicações numéricas demonstram a excelente eficiência do método EDMP. Os resultados indicam que EDMP é mais rápido para alcançar a solução ótima e que o algoritmo é altamente paralelizável. Além disso, visto que a comunicação entre os processadores é mínima, pode-se garantir grande diminuição no tempo de execução do programa em relação ao algoritmo sequencial.

Tabela 4.8: Valores ótimos considerando o método do Critério Global (Métrica L_{2R}).

Algoritmo	Volume [u.v.]	Destreza	Rigidez [u.r.]	$[d_3[u.l.]\ \ d_4[u.l.]\ \ r_2[u.l.]\ \ r_3[u.l.]]$	Tempo	N_{ava}
ED	1902,70	1,04	105,68	[3 3 3 0,1]	8,03h	150
	1903,30*	1,04*	105,68*	[3,00 3,00 3,00 0,11]*		
	0,74**	0,01**	0,01**	[0 0 0 0,01]**		
ECE 2 conjuntos	1905,10	1,14	105,90	[3,00 3,00 3,00 0,37]	3,05h	1462
	1844,85*	1,14*	103,51*	[2,95 2,97 2,97 0,39]*		
	132.71**	0.12**	5.19**	[0.15 0.07 0.09 0.39]**		
ECE 4 conjuntos	1908,20	1,11	105,82	[3,00 3,00 3,00 0,30]	7,24h	3468
	1805,18*	1,13*	102,17*	[2,95 2,94 2,90 0,34]*		
	137,30**	0,13**	5,13**	[0,11 0,12 0,14 0,34]**		
ECE 8 conjuntos	1903,40	1,08	105,71	[3,00 3,00 3,00 0,23]	14,23h	6800
	1832,20*	1,10*	103,23*	[2,98 2,96 2,91 0,26]*		
	142,31**	0,06**	5,15**	[0,06 0,12 0,21 0,30]**		
EDMP executado em 4 processadores	1902,76	1,03	105,67	[3,00 3,00 3,00 0,1]	45s	330
	1902,76*	1,03*	105,67*	[3,00 3,00 3,00 0,1]*		
	4e-13**	6e-16**	7e-14**	[0,0 0,0 0,0 4e-17]**		

* Média dos valores, ** desvio padrão e N_{ava} número de avaliações da função objetivo.

Tabela 4.9: Valores ótimos considerando o método da Ponderação dos Objetivos.

Coef.	Algoritmo	Volume [u.v.]	Destreza	Rigidez [u.r.]	$[d_3[u.c.]\ \ d_4[u.c.]\ \ r_2[u.c.]\ \ r_3[u.c.]]$	Tempo	N_{ava}
$w_1 = 0.05$ $w_2 = 0.90$ $w_3 = 0.05$	ED	1902,70 1703,20* 490,67**	1,04 1,04* 0,01**	105,68 97,11* 21,19**	[3,00 3,00 3,00 0,10] [2,87 2,82 2,80 0,10]* [0,61 0,48 0,65 0]**	15,45h	180
	ECE 2 conjuntos	1892,70 1164,10* 722,93**	1,04 1,07* 0,05**	105,42 72,17* 34,99**	[3,00 3,00 3,00 0,11] [1,93 2,47 2,70 0,19]* [1,23 0,60 0,50 0,15]**	4,78h	2278
	ECE 4 conjuntos	1896,30 1621,40* 374,60**	1,04 1,07* 0,05**	105,49 94,31* 16,70**	[3,00 3,00 3,00 0,11] [2,80 2,82 2,80 0,17]* [0,44 0,31 0,32 0,15]**	6,09h	2856
	ECE 8 conjuntos	1879,00 1635,90* 381,17**	1,04 1,06* 0,03**	104,88 94,68* 16,79**	[3,00 2,99 2,99 0,11] [2,81 2,82 2,84 0,13]* [0,33 0,31 0,30 0,05]**	11,11h	5304
	EDMP executado em 4 processadores	1902,75 1030,33* 873,13**	1,03 1,04* 0,02**	105,67 62,64* 43,13**	[3,00 3,00 3,00 0,1] [2,05 2,02 2,02 0,52]* [0,98 0,97 0,98 0,42]**	48s	360
$w_1 = 1/3$ $w_2 = 1/3$ $w_3 = 1/3$	ED	1902,70 1934,90* 99,03**	1,04 1,09* 0,18**	105,68 106,93* 3,87**	[3,00 3,00 3,00 0,10] [3,00 3,00 3,00 0,34]* [0 0 0 0,72]**	28,75h	900
	ECE 2 conjuntos	1902,60 1799,20* 143,27**	1,04 1,14* 0,20**	105,66 102,02* 5,26**	[3,00 3,00 3,00 0,10] [2,94 2,94 2,88 0,38]* [0,14 0,12 0,18 0,56]**	2,71h	1258
	ECE 4 conjuntos	1902,60 1863,30* 81,37**	1,04 1,13* 0,22**	105,67 104,45* 3,20**	[3,00 3,00 3,00 0,11] [2,94 2,98 2,94 0,39]* [0,12 0,05 0,12 0,74]**	6,81h	3128
	ECE 8 conjuntos	1902,60 1839,90* 105,97**	1,04 1,09* 0,12**	105,67 103,39* 3,91**	[3,00 3,00 3,00 0,11] [2,95 2,98 2,95 0,24]* [0,12 0,05 0,11 0,31]**	11,53h	5576
	EDMP executado em 4 processadores	2291,61 2289,68* 2,88**	1,71 1,71* 0,004**	120,75 120,72* 0,08**	[3,00 3,00 3,00 2,64] [3,00 3,00 3,00 2,64]* [0,0 0,0 0,0 0,007]**	43s	320

* Média dos valores, ** desvio padrão e N_{ava} número de avaliações da função objetivo.

4.3.2 Problema Restrito: Otimização de Sistemas Robóticos Considerando sua Topologia

De acordo com [36], para estudar as topologias dos manipuladores 3R é necessário conhecer as equações que as separam por meio das superfícies de singularidades. Serão considerados os manipuladores com três juntas rotacionais com eixos ortogonais ($\theta_2 = -90$ e $\theta_3 = 90$). O estudo deste tipo de manipulador se baseia na topologia das superfícies de singularidades do espaço de trabalho e é feito em função dos parâmetros de Denavit-Hartenberg restantes: d_2, d_3, d_4 e r_2. A fim de reduzir o número de parâmetros, sem perda de generalidade, podem-se normalizar todos estes parâmetros em relação à d_2 (obviamente qualquer outro parâmetro pode ser utilizado). Adotando-se $d_2 = 1$, os parâmetros geométricos considerados são d_3, d_4 e r_2. As três variáveis das juntas consideradas são: θ_1, θ_2 e θ_3.

Substituindo as hipóteses simplificadoras $\theta_2 = -90, \theta_3 = 90, r_3 = 0$ e $d_2 = 1$, na Eq.(4.3.36), obtém-se as seguintes expressões:

$$\begin{aligned} x &= \left[1 + (d_3 + d_4 c_3)\, c_2\right] c_1 - (r_2 + d_4 s_3)\, s_1 \\ y &= \left[1 + (d_3 + d_4 c_3)\, c_2\right] s_1 + (r_2 + d_4 s_3)\, c_1 \\ z &= -\,(d_3 + d_4 c_3)\, s_2 \end{aligned} \tag{4.3.51}$$

A existência de pelo menos um ponto de singularidade no espaço de trabalho com exatamente três soluções no Modelo Geométrico Inverso (MGI) coincidentes é difícil de ser mostrada diretamente do Modelo Geométrico Direto (MGD). A ideia é eliminar as variáveis de junta, θ_1 e θ_2, do sistema, a fim de obter uma condição sobre a última variável, θ_3. Para isto, são adicionadas as relações algébricas $c\theta_i^2 + s\theta_i^2 = 1, i = 1, 2, 3$ à Eq. (4.3.51) do MGD, resultando num sistema algébrico de 6 equações:

$$\begin{aligned} f_1 &= x - \left[1 + (d_3 + d_4 c_3)\, c_2\right] c_1 + (r_2 + d_4 s_3)\, s_1 \\ f_2 &= y - \left[1 + (d_3 + d_4 c_3)\, c_2\right] s_1 - (r_2 + d_4 s_3)\, c_1 \\ f_3 &= z + (d_3 + d_4 c_3)\, s_2 \\ f_4 &= c_1^2 + s_1^2 - 1 \\ f_5 &= c_2^2 + s_2^2 - 1 \\ f_6 &= c_3^2 + s_3^2 - 1 \end{aligned} \tag{4.3.52}$$

Agora, para eliminar variáveis θ_1 e θ_2 calcula-se a base de Groebner para o ideal $I = f_1, f_2, \cdots, f_6$. Para isso, usa-se um programa de álgebra computacional (Maple, Axiom ou Maxima). Assim, uma base de Groebner para I é composta pelos polinômios:

$$p_1(\theta_3) = c_3^2 + s_3^2 - 1 \tag{4.3.53}$$

e

$$p_2(\theta_3) = m_5 c_3^2 + m_4 s_3^2 + m_3 c_3 s_3 + m_2 c_3 + m_1 s_3 + m_0 \tag{4.3.54}$$

onde

$$\begin{cases} m_0 = -x^2 - y^2 + r_2^2 + \frac{(R+1-L)^2}{4} \\ m_1 = 2r_2 d_4 + (L - R - 1) d_4 r_2 \\ m_2 = (L - R - 1) d_3 d_4 \\ m_3 = 2r_2 d_3 d_4^2 \\ m_4 = d_4^2 (r_2^2 + 1) \\ m_5 = d_3^2 d_4^2 \end{cases} \tag{4.3.55}$$

com

$$R = x^2 + y^2 + z^2 \qquad \text{e} \qquad L = d_4^2 + d_3^2 + r_2^2 \tag{4.3.56}$$

Considerando a nova variável $t = tg(\theta_3/2)$ segue que:

$$c\theta_3 = \frac{1 - t^2}{1 + t^2} \qquad \text{e} \qquad s\theta_3 = \frac{2t}{1 + t^2} \tag{4.3.57}$$

Substituindo na Eq. (4.3.54) e realizando algumas manipulações algébricas, obtém-se o seguinte polinômio:

$$P(t) = at^4 + bt^3 + ct^2 + dt + e \tag{4.3.58}$$

onde

$$\begin{aligned} a &= m_5 - m_2 + m_0 \\ b &= -2m_3 + 2m_1 \\ c &= -2m_5 + 4m_4 + 2m_0 \\ d &= 2m_3 + 2m_1 \\ e &= m_5 + m_2 + m_0 \end{aligned} \tag{4.3.59}$$

Um manipulador é cuspidal se o polinômio $P(t)$, que possui grau 4 e coeficientes que dependem de x, y, z, d_4, d_3 e r_2, admitir raiz real tripla. Isto equivale a resolver o seguinte sistema:

$$S(t, a, b, c, d, e) = \begin{cases} P(t, Z, R, d_3, d_4, r_2) = 0 \\ \frac{\partial P}{\partial t}(t, Z, R, d_3, d_4, r_2) = 0 \\ \frac{\partial^2 P}{\partial t^2}(t, Z, R, d_3, d_4, r_2) = 0 \end{cases} \tag{4.3.60}$$

onde $Z = z^2$. No sistema polinomial dado pela Eq. (4.3.60), as variáveis do problema são t, R e Z, enquanto os parâmetros estritamente positivos do problema são d_3, d_4 e r_2 (observe que o anulamento de um destes parâmetros resulta em um manipulador não cuspidal). Para escrever a condição dependendo somente dos parâmetros DH é necessário eliminar as variáveis t, R e Z do sistema da pela Eq. (4.3.60). Resolver o sistema é equivalente a dividir o primeiro quadrante do espaço dos parâmetros (o espaço $d_3 d_4 r_2$) em várias regiões em que o número das soluções do problema é constante. Para cada região, são obtidos o número de soluções do problema e um conjunto de parâmetros do

manipulador que representam esta região. De acordo com as hipóteses $Z > 0$ e $R - Z > 0$, é possível eliminar as variáveis t, Z e R. Depois de várias mudanças de variáveis e utilizando a base de Grobner, segundo [37], são obtidas as cinco equações abaixo:

$$\begin{aligned}
&g_1 : r_2^2 + d_3^2 - d_4^2 = 0,\\
&g_2 : d_4^2\left[(1 + r_2^2 + d_3^2)^2 - 4d_3^2\right](r_2^2 + d_3^2 - d_4^2) - r_2^2 d_3^2 = 0,\\
&g_3 : r_2^2 + d_4 - d_3 + d_4 = 0,\\
&g_4 : (d_3 - 1)^2 (d_3^2 - d_4^2) + r_2^2 d_3^2 = 0,\\
&g_5 : (d_3 + 1)^2 (d_3^2 - d_4^2) + r_2^2 d_3^2 = 0.
\end{aligned} \tag{4.3.61}$$

Agora tem-se condições de obter a superfície que distingue dois tipos de manipuladores: binário e quaternário. O manipulador é binário (resp. quaternário) se o MGI tem no máximo 2 (resp. 4) soluções. Um manipulador binário é sempre não cuspidal. Uma vez que a equação para g_2 tem grau 2 em d_4 é possível obter a equação de separação:

$$C_1 : d_4 = \sqrt{\frac{1}{2}\left(d_3^2 + r_2^2 - \frac{(d_3^2 + r_2^2)^2 - d_3^2 + r_2^2}{AB}\right)}, \tag{4.3.62}$$

onde $A = \sqrt{(d_3 + 1)^2 + r_2^2}$ e $B = \sqrt{(d_3 - 1)^2 + r_2^2}$.

As outras superfícies são obtidas anulando o determinante da matriz jacobiana J do Modelo Geométrico Direto:

$$det(J) = d_4(d_3 + d_4 c_3)[s_3 + (d_3 s_3 - r_2 c_3)c_2] \tag{4.3.63}$$

Assim, as superfícies de separação de C_2, C_3 e C_4 são obtidas resolvendo a Eq. (4.3.63) e são dadas por:

$$C_2 : d_4 = d_3/(1 + d_3)\sqrt{(d_3 + 1)^2 + r_2^2} \tag{4.3.64}$$

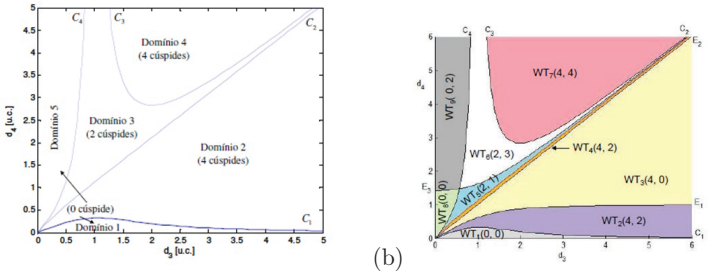
$$C_3 : d_4 = d_3/(d_3 - 1)\sqrt{(d_3 - 1)^2 + r_2^2}, \quad \text{com } d_3 > 1 \tag{4.3.65}$$

$$C_4 : d_4 = d_3/(1 - d_3)\sqrt{(d_3 - 1)^2 + r_2^2} \quad \text{com } d_3 < 1 \tag{4.3.66}$$

Resumindo, o espaço dos parâmetros (d_3, d_4 e r_2) de um manipulador ortogonal 3R está dividido em cinco domínios separados pelas superfícies C_1, C_2, C_3 e C_4, definidas, respectivamente, pelas Eqs. (4.3.62), (4.3.64), (4.3.65) e (4.3.66).

A Fig. 4.17a mostra as curvas de separação de uma secção plana (d_3, d_4) do espaço dos parâmetros, resultando em cinco domínios, adotando um valor fixo para $r_2 = 1$. A Fig. 4.17b mostra o espaço de parâmetros divididos de acordo com o número de pontos de cúspides e de pontos de nós. Os domínios, de acordo com o número de pontos de

cúspide, são divididos em subdomínios que contêm o mesmo número de nós. Cada sub-domínio define a topologia do espaço de trabalho indicada por $WT_i(\alpha, \beta)$, onde α representa o número de pontos de cúspide e β o número de pontos de nós.



(a) (b)

Figura 4.17: Divisão do espaço de parâmetros considerando $r_2 = 1$: (a) De acordo com a superfície de separação de topologias, (b) De acordo com o número de pontos de cúspides e nós. Fonte: [6] pg. 91 e pg. 102.

A Eq. (4.3.62) representa a superfície de separação entre os manipuladores binário e quaternário. Os manipuladores que pertencem ao domínio 1 são binários, tem uma cavidade toroidal na sua área de trabalho e não têm pontos de cúspide e de nó. Este manipulador, representado na Fig. 4.18a, caracteriza o primeiro tipo de manipulador, cuja topologia é conhecida como $WT_1(0, 0)$.

O domínio 2 representa os manipuladores quaternários que têm 4 pontos de cúspide. Esta região pode ser subdividida em três subdomínios através das superfícies E_1 e E_2. A topologia do espaço de trabalho $WT_2(4, 2)$, representada na Fig. 4.18b, tem quatro pontos de cúspide, 2 nós, uma cavidade toroidal, duas regiões com 4 soluções e uma região com 2 soluções em MGI. A topologia $WT_3(4, 0)$ contém manipuladores com 4 pontos de cúspides, zero nós, nenhuma cavidade toroidal, uma região com 4 soluções e outra com 2 soluções em MGI, como ilustrado na Fig. 4.18c. A transição entre as topologias WT_2 e WT_3 é o limite entre os manipuladores que contêm uma cavidade toroidal em seu espaço de trabalho e aqueles que não o contêm. Segundo [6], a superfície de separação entre as topologias é dada pela expressão:

$$E_1 : d_4 = 0,5(A - B), \tag{4.3.67}$$

onde A e B são dados na Eq. (4.3.62).

No domínio 2, ainda é possível caracterizar a topologia representada na Fig. 4.18d, denotada por $WT_4(4, 2)$, que contém 4 pontos de cúspide e 2 nós. Estes nós são diferentes dos nós de WT_2 uma vez que não delimitam uma cavidade toroidal mas uma região de quatro soluções

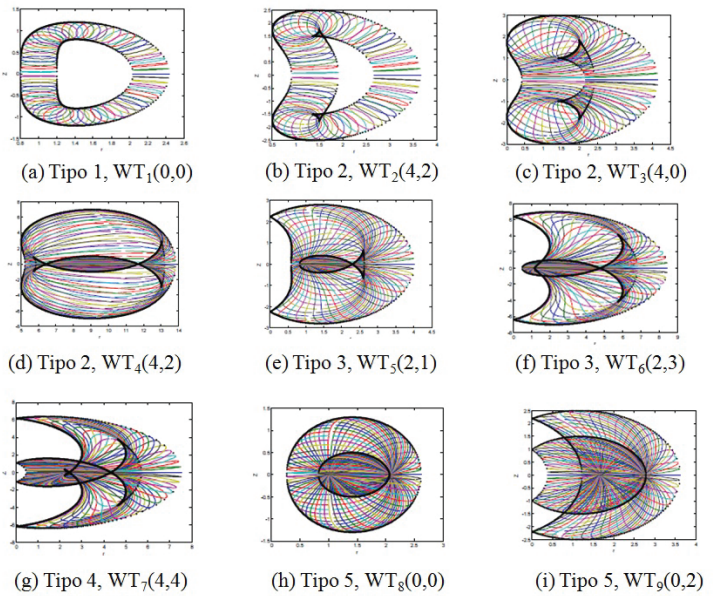


Figura 4.18: Seção radial de manipuladores ortogonais 3R, mostrando os 5 tipos de manipuladores. Fonte: [36].

no MGI. Neste caso, a superfície de separação E_2, entre as topologias WT_3 e WT_4 é definida por:

$$E_2 : d_4 = d_3. \tag{4.3.68}$$

A superfície C_2 separa os domínios 2 e 3 e é dada pela Eq. (4.3.64). O domínio 3 é composto por manipuladores que apresentam dois pontos de cúspides no invólucro interno e pode ser dividido em dois subdomínios através da superfície E_3. Os manipuladores descritos por $WT_5(2,1)$ tem dois pontos de cúspides no invólucro interno, um ponto de nó e a forma de um peixe, como mostrado na secção radial apresentada na Fig. 4.18e. Além disso, a Fig. 4.18f apresenta a secção radial de um manipulador que pertence a topologia $WT_6(2,3)$, que tem 2 pontos de cúspide e 3 nós. A Eq. (4.3.71) define a superfície de separação entre as topologias WT_5 e WT_6. Além disso, esta superfície também separa a topologia dos espaços de trabalho WT_8 e WT_9 que estão contidas no domínio 5.

$$E_3 : d_4 = 0,5(A + B). \tag{4.3.69}$$

A superfície C_3, Eq. (4.3.65), separa os manipuladores dos domínios 3 e 4. No domínio 4 , os manipuladores são do tipo 4, representado por

$WT_7(4,4)$, possuem 4 pontos de cúspide e 4 nós, como pode ser visto na Fig. 4.18g. Os 4 pontos de cúspide são compartilhados entre as superfícies internas e externas de singularidade.

Finalmente, o domínio 5 corresponde aos manipuladores que não possuem pontos de cúspide (Figs. 4.18h e 4.18i). No caso do domínio 5, ao contrário dos manipuladores do tipo 1, o invólucro interno não é definido por uma cavidade toroidal, mas por uma região com 4 soluções em MGI. A superfície C_4, representada pela Eq. (4.3.66), separa os domínios 3 e 5.

O domínio 5 corresponde aos manipuladores do tipo 5. Esta região é dividida em 2 subdomínios por meio da superfície E_3. Na Fig. 4.18h, a topologia representada por $WT_8(0,0)$ não tem pontos de cúspides ou nós. Como mencionado anteriormente, o invólucro interno não é definido por uma cavidade toroidal, mas por uma região com 4 soluções no MGI. Finalmente, a Fig. 4.18i apresenta um manipulador que pertence a topologia $WT_9(0,2)$, com 0 pontos de cúspide e 2 pontos de nós obtidos pela intersecção do invólucro interno e externo.

O problema de otimização é formulado com o objetivo de obter os parâmetros geométricos ideais do manipulador 3R para maximizar o espaço de trabalho e da rigidez do sistema e para minimizar a destreza para que as topologias especificadas pelo projetista sejam obedecidas. Visto que o problema se trata de um problema de otimização multiobjetivo é necessário localizar todas as compensações possíveis entre várias funções objetivos que geralmente são conflitantes entre si. As restrições dependem da topologia escolhida para o robô, de acordo com a Fig. 4.18. A otimização será investigada utilizando Algoritmos Genéticos (GA), Evolução Diferencial (DE) e Evolução Diferencial Melhorada implementado em paralelo (EDMP).

Os algoritmos evolutivos foram desenvolvidos para problemas sem restrições. Assim, no caso de problemas de otimização com restrições, é necessário introduzir modificações no método. Será utilizado o conceito de Função de Penalidade de [35]. Foi adotado $r_p = 1000$. Então, o problema pode ser reescrito como se segue:

$$\begin{array}{c} \text{Max} \;\; F(x) = f(x) + r_p P(x), \\ \text{onde} \;\; f(x) = [V, D, R] \\ \text{e} \;\; P(x) = \max(0, g_j(x))^2. \end{array} \tag{4.3.70}$$

Sujeito à : $g_j(x) \leq 0;\ j = 1, \cdots, k$ e $x^l \leq x_i \leq x^u,\ i = 1, 2, 3$.

Os parâmetros geométricos são as variáveis de projeto dadas por $x = (d_3, d_4, r_2)^T$. Os limites inferiores e superiores adotados para o comprimento do braço (restrições laterais) são: $0,1 \leq x_i \leq 3,0,\ i = 1, 2, 3$.

Nesta simulação, dois métodos de otimização multiobjetivo são utilizados: Método dos Objetivos Ponderados e Método do Critério Global

(métrica L_{2r} e métrica L_{3r}) apresentado por [39].

A estratégia de soma ponderada converte o vetor $f(x)$ de um problema multiobjetivo em um problema de otimização escalar através de uma soma ponderada de todos os objetivos como Eq. (4.3.71). Os coeficientes de ponderação w_i representam a importância relativa de cada critério. Assim,

$$\text{Maximizar } F(x) = w_1 Vc_1 - w_2 Dc_2 + w_3 Rc_3 - r_p P(x), \text{ onde } \sum_{i=1}^{3} w_i = 1, \tag{4.3.71}$$

sendo que o volume do espaço de trabalho V é dada pela Eq. (4.3.41), a rigidez R é calculada utilizando a Eq. (4.3.49) e a destreza D é representada na Eq.(4.3.50).

A ponderação dos objetivos é o modelo substituto mais comum para problemas de otimização vetoriais. A dificuldade aqui é anexar os coeficientes de ponderação para cada um dos objetivos. Os coeficientes de ponderação não necessariamente correspondem diretamente à importância relativa dos objetivos ou permitem que os *tradeoffs* entre os objetivos sejam expressos. Para que w_i na Eq. (4.3.71) possa refletir estreitamente a importância de objetivos todas as funções devem ser expressas em unidades que aproximadamente representam os mesmos valores numéricos. Os melhores resultados são geralmente obtidos quando $c_i = 1/f_i^0$, onde f_i^0 representa a solução ideal, que indica o valor mínimo de cada i-ésima função. Para determinar esta solução, deve-se encontrar o mínimo possível para todas as funções objetivo separadamente. Neste caso, o vetor $f^0 = [V_{id}, D_{id}, R_{id}]^T$ é o vetor dos valores ideais de um problema de otimização multiobjetivo.

No Método do Critério Global, o problema de otimização multiobjetivo é transformado em um problema de otimização escalar usando um critério global. A função que descreve este critério global pode ser definida como uma possível solução próxima da solução ideal encontrada. Neste caso, a métrica-L_{2r} e a métrica-L_{3r}, são dadas respectivamente por:

$$\text{Minimizar } F(x) = \left(\left(\frac{V_{id}-V}{V_{id}} \right)^2 + \left(\frac{D_{id}-D}{D_{id}} \right)^2 + \left(\frac{R_{id}-R}{R_{id}} \right)^2 \right)^{1/2} + r_p P(x). \tag{4.3.72}$$

$$\text{Minimizar } F(x) = \left(\left| \frac{V_{id}-V}{V_{id}} \right|^3 + \left| \frac{D_{id}-D}{D_{id}} \right|^3 + \left| \frac{R_{id}-R}{R_{id}} \right|^3 \right)^{1/3} + r_p P(x). \tag{4.3.73}$$

Para este problema foram adotados os seguintes parâmetros:

- ED: número de indivíduos na população $N_p = 15$; 100 gerações, fator de diferença ponderada $F = 0{,}8$ e probabilidade de cruzamento $CR = 0{,}5$.

- AG: $N_p = 80$ indivíduos, 100 gerações, probabilidades de cruzamento e mutação: 0,60 e 0,02.

- EDMP: população com 64 indivíduos divididos em 4 processadores, 100 gerações, fator de diferença $F = 0{,}8$ e probabilidade de cruzamento $CR = 0{,}6$. O critério de parada do algoritmo EDMP adotado foi o número máximo de gerações da população e a verificação de sua estagnação. O processo de optimização é interrompido quando não ocorre uma melhoria significativa no valor da função objetivo após 30 iterações. Isto é, se a seguinte comparação $|f_k - f_{k+1}| < 10^{-6}$ ocorre 30 vezes e as 100 iterações ainda não foram completadas, o algoritmo para. As simulações foram desenvolvidas em um computador Intel® Core^{TM} i5-430M e 6 GB de RAM.

Exemplo 1 - Manipuladores da Topologia $WT_1(0,0)$

Neste exemplo, será considerado uma aplicação onde o manipulador deve pertencer a topologia WT_1 (ver Fig. 4.18b). Neste caso, as seguintes restrições laterais devem ser obedecidas: $0{,}1 < d_3, d_4, r_2 < 3{,}0$ [u.c.].

Os pontos pertencem à curva C_1, dada pela Eq. (4.3.62). As soluções ideais calculadas usando ED foram: $V_{id} = 99{,}7175$ [u.v.]; $D = 1{,}1979$ e $R_{id} = 35{,}8325$ [u.r.].

Os resultados ótimos obtidos através do Método da Ponderação dos Objetivos, Eq. (4.3.71), são apresentados na Tab. 4.10. Observando esta tabela, pode-se notar que a melhor solução depende do interesse do projetista, pois cada função objetivo está conflitando com a outra. Neste exemplo, quando se adotou os coeficientes de ponderação igual a 0,8 para o volume (w_1) ou para a rigidez (w_3), obteve-se resultados semelhantes. Isto é devido ao fato de que ambos são maximizados e apresentaram um comportamento semelhante. Mas ao adotar o coeficiente de ponderação igual a 0,8 para a destreza (w_2) observou-se que um resultado diferente foi obtido e a destreza foi significativamente melhorada. Os resultados indicam que este problema é sensível ao valor da destreza. Quando esta função é priorizada, os valores ótimos do volume e da rigidez são fortemente modificados.

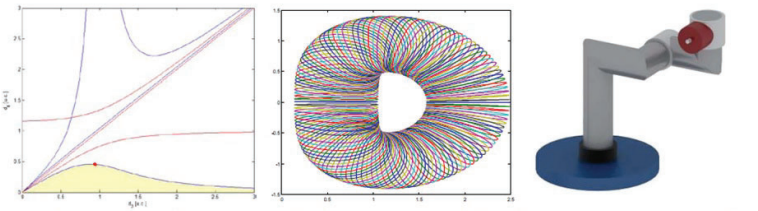
Os valores calculados usando o Método do Critério Global, dada nas Eqs. (4.3.72) e (4.3.73), podem ser observados na Tab. 4.11. Nesta técnica, a ideia consiste em minimizar o erro relativo das funções em

relação aos valores ideais. As soluções obtidas representam um compromisso entre as três funções objetivos.

Os valores encontrados com os algoritmos AG e ED apresentados nas Tabs. 4.10 e 4.11 são de [36].

Observando as Tabs. 4.10 e 4.11 pode-se observar que ambas as técnicas encontraram a solução ótima. Porém, os resultados divergiram um pouco entre as técnicas. Os valores encontrados com os algoritmos AG e ED são de [36] e não foram verificados nessa pesquisa.

Considerando a métrica-L_{2r}, o ponto ótimo obtido pelo método Evolução Diferencial está marcado na Fig. 4.19a. A área da secção radial ótima do espaço de trabalho é apresentado na Fig. 4.19b. Comparando a secção radial da Fig. 4.19b com a Fig. 4.18a, pode-se notar que os parâmetros projetados resultam em um manipulador com um volume maior (o vazio da área de trabalho foi reduzida). O manipulador ótimo pertencente a topologia $WT_1(d_2 = 1,\ r_3 = 0)$ é apresentado na Fig. 4.19c.



(a) Espaço de parâmetros com d_2=1, r_3=0. (b) Área da secção radial do espaço de trabalho. (c) Esquema de um robô 3R.

Figura 4.19: Projeto ótimo de um robô 3R considerando a métrica-L_{2r} pela Evolução Diferencial - Exemplo 1.

Tabela 4.10: Valores ótimos considerando o método da Ponderação dos Objetivos para o exemplo 1.

Coef. de Poderação w_i	Algoritmo	$[d_3 \quad d_4 \quad r_2]$ [u.c.]	Volume [u.v.]	Destreza	Rigidez [u.r.]	Tempo (min)	N_{ava}
$w_1 = 0,33$ $w_2 = 0,33$ $w_3 = 0,33$	AG$^{\Delta}$	0,65 0,44 0,52	25,46	1,19	6,05	117,7	
	ED$^{\Delta}$	0,97 0,73 0,10	58,63	1,87	10,72	67,34	
	EDMP	0,92 0,45 0,61	39,18	1,41	8,15		
		0,94* 0,45* 0,61*	39,67*	1,42*	8,23*	0,55	656
		0,02** 0,001** 0,002**	0,85**	0,03**	0,14**		
$w_1 = 0,10$ $w_2 = 0,80$ $w_3 = 0,10$	AG$^{\Delta}$	0,60 0,43 0,49	22,64	1,18	5,63	89,12	
	ED$^{\Delta}$	0,60 0,43 0,48	22,68	1,18	5,63	60,88	
	EDMP	0,70 0,44 0,56	28,17	1,21	6,49		
		0,67* 0,44* 0,54*	26,70*	1,21*	6,26*	0,54	736
		0,02** 0,001** 0,01**	1,00**	0,004**	0,15**		
$w_1 = 0,10$ $w_2 = 0,10$ $w_3 = 0,80$	AG$^{\Delta}$	1,00 0,44 0,61	42,57	1,51	8,65	121,9	
	ED$^{\Delta}$	0,97 0,74 0,10	59,14	1,89	10,79	56,39	
	EDMP	0,93 0,79 0,10	59,93	1,99	11,01		
		1,01* 0,61* 0,34*	52,37*	1,86*	10,09*	0,52	704
		0,07** 0,18** 0,24**	7,57**	0,13**	0,91**		

$^{\Delta}$ Fonte: [36], * Média dos valores, ** desvio padrão e N_{ava} número de avaliações da função objetivo.

Tabela 4.11: Valores ótimos considerando o método do Critério Global para o exemplo 1.

Métrica	Algoritmo	$[d_3 \quad d_4 \quad r_2]$ [u.c.]	Volume [u.v.]	Destreza	Rigidez [u.r.]	Tempo (min)	N_{ava}
L_{2r}	AG$^{\Delta}$	0,97 0,45 0,59	41,73	1,45	8,46	87,39	
	ED$^{\Delta}$	0,95 0,45 0,59	40,87	1,42	8,35	65,17	
	EDMP	0,84 0,45 0,61	35,05	1,30	7,51		
		0,83* 0,45* 0,60*	34,50*	1,29*	7,43*	0,57	736
		0,01* 0,001* 0,002*	0,54**	0,001**	0,07**		
L_{2r}	AG$^{\Delta}$	0,92 0,54 0,41	44,33	1,67	8,66	75,45	
	ED$^{\Delta}$	0,97 0,73 0,10	58,64	1,87	10,72	41,52	
	EDMP	0,76 0,45 0,58	31,15	1,24	6,93		
		0,75* 0,45* 0,58*	30,54*	1,23*	6,84*	0,52	672
		0,01* 0,002* 0,006*	0,49**	0,005**	0,07**		

$^{\Delta}$ Fonte: [36], * Média dos valores, ** desvio padrão e N_{ava} número de avaliações da função objetivo.

Exemplo 2 - Manipuladores da Topologia $WT_3(4,0)$

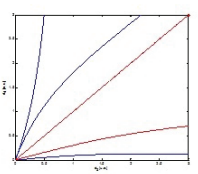
Agora, considerando um manipulador que pertence a topologia WT_3, são adotadas as seguintes restrições: $0{,}1 < d_3, d_4, r_2 < 3{,}0$ [u.c.]. Pontos acima da curva E_1, dada pela Eq. (4.3.67) e pontos abaixo da curva E_2, dada pela Eq. (4.3.68). As soluções ideais calculadas utilizando ED foram: V_{id}= 1896,784 [uv]; D_{id} = 1,018 e R_{id} = 105,66 [us]. Para este caso, os melhores resultados obtidos pelo Método da Ponderação dos Objetivos são mostrados na Tab. 4.12 e a Tab. 4.13 apresenta as soluções ótimas calculadas usando o Método do Critério Global. Como observado no Exemplo 1, a melhor solução depende de interesse do projetista.

Considerando-se a Métrica L_{2r}, o ponto ótimo obtido por Evolução Diferencial é apresentado na Fig. 4.20a. O esquema do manipulador ótimo pertencente a topologia WT_3 está ilustrado na Fig. 4.20b.

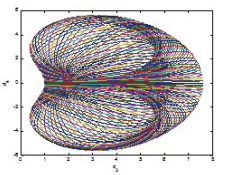
A área da secção radial ótima do espaço de trabalho é apresentado na Fig. 4.20b. Comparando a secção radial da Fig. 4.20b com a Fig. 4.18c pode-se observar que o espaço de trabalho é maior. Além disso, os manipuladores deste tipo de topologia permanecem com 4 pontos de cúspides, uma região com 4 soluções e uma outra com 2 soluções em Modelo Geométrico Inverso.

No exemplo 1, observe que os baixos valores encontrados em relação aos valores ideais, são devido às dificuldades impostas pela restrição C_1. No exemplo 2, os valores não sofrem muitas alterações devido às restrições serem mais simples.

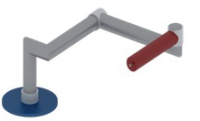
É importante observar que as metodologias aplicadas foram eficazes para obter as melhores soluções obedecendo as restrições de topologia.



(a) Espaço de parâmetros com d_2=1, r_3=0.



(b) Área da secção radial do espaço de trabalho.



(c) Esquema de um robô 3R.

Figura 4.20: O projeto ideal de um robô 3R, considerando Métrica L_{2r} usando Evolução Diferencial - Exemplo 2.

Novamente, o algoritmo EDMP mostrou-se muito eficiente. Os resultados mostram que EDMP é mais rápido na busca pela solução ótima. Entre os métodos de otimização discutidos aqui, a saber, Algoritmo Genético, Evolução Diferencial e Evolução Diferencial Melhorada implementado em paralelo, a metodologia EDMP apresentou os melho-

Tabela 4.12: Valores ótimos considerando o método da Ponderação dos Objetivos para o exemplo 2.

Coef. de Poderação w_i	Algoritmo	$[d_3 \quad d_4 \quad r_2]$ [u.c.]	Volume [u.v.]	Destreza	Rigidez [u.r.]	Tempo (min)	N_{ava}
$w_1 = 0,33$ $w_2 = 0,33$ $w_3 = 0,33$	AG$^{\Delta}$	3,00 3,00 3,00	1896,78	1,018	105,66	55,03	
	ED$^{\Delta}$	3,00 3,00 3,00	1896,78	1,018	105,66	12,87	
	EDMP	3,00 3,00 3,00	1887,62	1,02	105,66		
		3,00* 3,00* 3,00*	1887,62*	1,02*	105,66*	0,43	560
		0** 0** 0**	1,3e-12**	1,3e-15**	2,8e-14**		
$w_1 = 0,80$ $w_2 = 0,10$ $w_3 = 0,10$	AG$^{\Delta}$	3,00 3,00 3,00	1896,78	1,018	105,66	114,02	
	ED$^{\Delta}$	3,00 3,00 3,00	1896,78	1,018	105,66	66,37	
	EDMP	3,00 3,00 3,00	1887,62	1,018	105,66		
		3,00* 3,00* 3,00*	1887,62*	1,02*	105,66*	0,41	528
		0** 0** 0**	1,3e-12**	1,3e-15**	2,8e-14**		
$w_1 = 0,10$ $w_2 = 0,80$ $w_3 = 0,10$	AG$^{\Delta}$	3,00 2,97 2,94	1850,93	1,010	104,12	114,56	
	ED$^{\Delta}$	3,00 3,00 3,00	1892,50	1,017	105,59	9,80	
	EDMP	3,00 3,00 3,00	1887,63	1,017	105,66		
		3,00* 3,00* 3,00*	1887,62*	1,02*	105,66*	0,40	512
		0** 0** 0**	1,3e-12**	1,3e-15**	2,8e-14**		
$w_1 = 0,10$ $w_2 = 0,10$ $w_3 = 0,80$	AG$^{\Delta}$	3,00 3,00 3,00	1896,78	1,018	105,66	119,84	
	ED$^{\Delta}$	3,00 3,00 3,00	1896,78	1,018	105,66	55,21	
	EDMP	3,00 3,00 3,00	1887,62	1,018	105,66		
		3,00* 3,00* 3,00*	1887,62*	1,02*	105,66*	0,40	528
		0** 0** 0**	1,3e-12**	1,3e-15**	2,8e-14**		

$^{\Delta}$ Fonte: [36], * Média dos valores, ** desvio padrão e N_{ava} número de avaliações da função objetivo.

Tabela 4.13: Valores ótimos considerando o método do Critério Global para o exemplo 2.

Métrica	Algoritmo	$[d_3 \;\; d_4 \;\; r_2]$ [u.c.]	Volume [u.v.]	Destreza	Rigidez [u.r.]	Tempo (min)	N_{ava}
L_{2r}	AG$^{\Delta}$	3.00 3.00 3.00	1896.78	1.018	105.66	59.78	
	ED$^{\Delta}$	3.00 3.00 3.00	1896.78	1.018	105.66	13.35	
	EDMP	3.00 3.00 2.98	1877.21	1.016	105.48		
		3.00* 3.00* 2.99*	1880.76*	1.02*	105.51*	0.40	512
		0** 0.002** 0.01**	7.25**	0.001**	0.17**		
L_{3r}	AG$^{\Delta}$	3.00 3.00 3.00	1896.77	1.018	105.66	66.28	
	ED$^{\Delta}$	3.00 3.00 3.00	1896.78	1.018	105.66	11.93	
	EDMP	3.00 2.96 2.93	1833.32	1.009	103.82		
		3.00* 2.95* 2.93*	1831.42*	1.01*	103.74*	0.43	528
		0.002** 0.003** 0.002**	4.78**	9.7e-05**	0.17**		

$^{\Delta}$ Fonte: [36], * Média dos valores, ** desvio padrão e N_{ava} número de avaliações da função objetivo.

res resultados levando em conta o esforço computacional, uma vez que foram necessários apenas alguns segundos para obter a solução ótima, enquanto os outros métodos precisaram de minutos ou de horas para encontrá-la.

4.4 Exercícios propostos

Os seguintes exercícios são atividades propostas pelos autores a fim de que o leitor possa modelá-los e resolvê-los com o algoritmo de otimização Evolução Diferencial.

Nos seguintes sites [15], [32] e [25] o leitor encontrará o código computacional do ED escrito em Matlab, Java e Python, respectivamente.

Exemplo 4.1. *(Retirado de [46]) Uma empresa de aço tem uma rede de distribuição conforme a Fig. 4.1. Duas minas M1 e M2 produzem 40t e 60t de mineral de ferro, respectivamente, que são distribuídos para dois estoques intermediários S1 e S2. A planta de produção P tem uma demanda de 100t de mineral de ferro. As vias de transporte têm limites de toneladas de mineral de ferro que podem ser transportadas e custos de transporte por toneladas de mineral de ferro (veja Fig. 4.1). A direção da empresa quer determinar a transportação que minimiza os custos.*

Considerações para a resolução: *trata-se de um problema linear, de seis variáveis reais, uniobjetivo e com restrições, por isso, o método da penalidade deverá ser utilizado ao modelar o problema para ser resolvido com ED. Note que esse problema também pode ser resolvido por meio da programação linear utilizando, por exemplo, o algoritmo Simplex.*

Exemplo 4.2. *(Retirado de [46]) Um fazendeiro está estudando a divisão de sua propriedade nas seguintes atividades produtivas:*

a) Arrendamento - Destinar certa quantidade de alqueires Para a plantação de cana de açúcar, a uma usina local, que se encarrega da atividade e paga pelo aluguel da terra R$ 300,00 por alqueire por ano;

b) Pecuária - Usar outra parte para a criação de gado de corte. A recuperação das pastagens requer adubação (100 kg/alqueire) e irrigação(100.000 litros de água/alqueire) por ano. O lucro estimado nessa atividade é de R$400,00 por alqueire por ano.

c) Plantio de Soja - Usar uma terceira parte para o plantio de soja. Essa cultura requer 200 kg por alqueire de adubos e 200.000 litros de água por alqueire para irrigação por ano. O lucro estimado nessa atividade é de R$500,00 por alqueire por ano.

A disponibilidade de recursos por ano é de 12.750.000 litros de água,14.000 kg de adubo e 100 alqueires de terra. Quantos alqueires

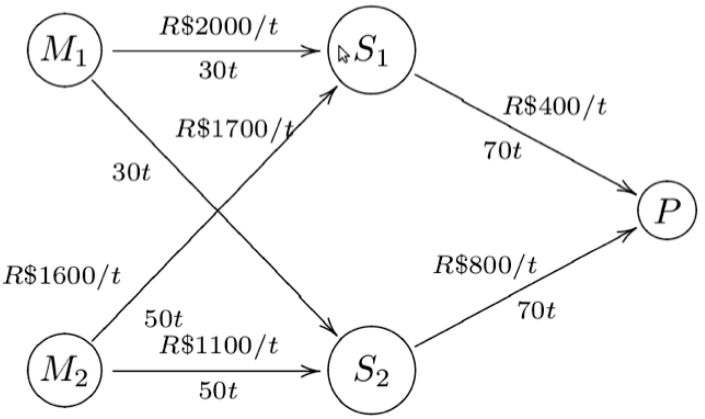


Figura 4.21: Rede de distribuição de uma empresa de aço.

deverá destinar a cada atividade para proporcionar o melhor retorno? Construa o modelo de decisão.

Considerações para a resolução: *trata-se de um problema linear, de três variáveis reais, uniobjetivo e com restrições, por isso o método da penalidade deverá ser utilizado ao modelar o problema para ser resolvido com ED. Note que esse problema também pode ser resolvido por meio da programação linear utilizando, por exemplo, o algoritmo Simplex.*

Exemplo 4.3. *(Retirado de [12]) Sob a ação de uma força F, o pendulo move de A até a posição de equilíbrio B, conforme Figura 4.22. Obtenha esta posição de equilíbrio, que corresponde à mínima energia potencial:*

$$min \;\; Ep = Wy - Fx,$$

onde, $x = Lsen\theta$, $y = L(1 - cos\theta)$.

Dados: $W = 500N$; $F = 100N$; $L = 2,5m$, restrições laterais: $\theta \in [0, \pi/2]$.

Considerações para a resolução: *trata-se de um problema não-linear, de duas variáveis reais, uniobjetivo e apenas com restrições laterais, e, por isso, o método da penalidade não precisa ser aplicado ao resolver esse problema por meio do algoritmo da ED. As restrições laterais entram em forma de dados no algoritmo.*

Exemplo 4.4. *(Retirado de [12]) Determinar a posição de equilíbrio estático de um sistema constituído de 2 molas (K_1 e K_2) solicitado por*

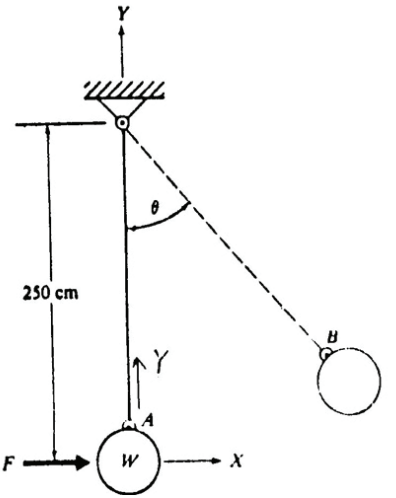


Figura 4.22: Equilíbrio estático de um pendulo submetido a uma força F.

duas forças constantes (P_1 e P_2), conforme Figura 4.23, de forma a minimizar sua energia potencial:

$$min \;\; Ep = \;\; 0,5k_1 \left(\sqrt{X_1^2 + (l_1 - X_2)^2} - l_1 \right)^2 + \\ +0,5k_2 \left(\sqrt{X_1^2 + (l_2 - X_2)^2} - l_2 \right)^2 - P_1 X_1 - P_2 X_2$$

Dados: $P_1 = P_2 = 5$ *N;* $K_1 = 8$ *N/cm;* $K_2 = 1$ *N/cm;* $l_1 = l_2 = 10$ *cm; restrições laterais:* $X_i \in [0, 10]$.

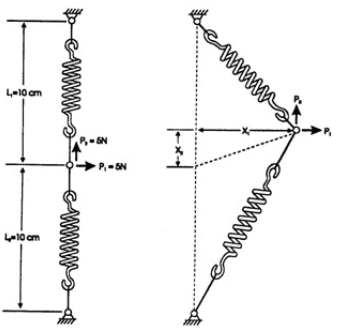


Figura 4.23: Equilíbrio estático de um sistema de 2 molas.

Considerações para a resolução: *trata-se de um problema não-linear, de duas variáveis reais, uniobjetivo e apenas com restrições laterais, e, por isso, o método da penalidade não precisa ser aplicado.*

Exemplo 4.5. *(Retirado de [12]) Deseja-se projetar um mastro de bandeira de altura H com a menor massa possível obedecendo certas restrições impostas ao problema. O mastro é composto por um tubo circular, oco e uniforme, sendo d_o e d_i seus diâmetros externo e interno, respectivamente. O projeto deve considerar a ação de fortes ventos que eventualmente pode atuar sobre o mastro da bandeira, que deve resistir sem apresentar falhas. Por razões de projeto, o mastro será considerado com uma viga engastada sujeita a uma carga uniformemente distribuída na lateral w (kN/m), representando a atuação do vento. Além disso, o vento também induz uma carga concentrada P (kN) no topo do mastro, como mostrado na Figura 4.24.*

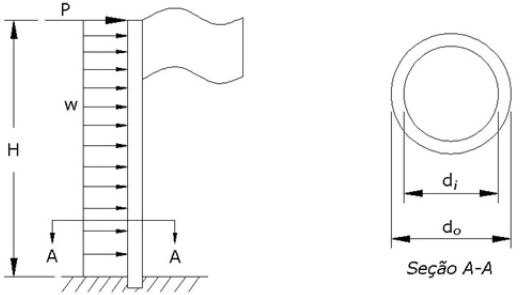


Figura 4.24: Projeto de um mastro de bandeira.

O problema se resume a minimização da área da seção transversal do mastro, definida pela função A, aliada a algumas restrições: a deflexão no topo δ não deve exceder 10 cm; a razão do diâmetro médio pela espessura não deve exceder 60; as espessuras mínima e máxima são 0,8 e 2 cm, respectivamente; a tensão de flexão admissível é σ_b=165 MPa e a tensão de cisalhamento admissível é τ_s=50 MPa.

$$A = \frac{\pi}{4}\left(d_o^2 - d_i^2\right).$$

As funções de restrição são dadas por:

$\delta = \frac{PH^3}{3EI} + \frac{wH^4}{8EI} \leq 10\ cm$
$\frac{d_o+d_i}{d_o-d_i} \leq 60$
$0,8 \leq \frac{d_o-d_i}{2} \leq 2\ cm$
$\sigma = \frac{M}{2I} d_o \leq 165\ MPa$
$\tau = \frac{S}{12I}\left(d_o^2 + d_o d_i + d_i^2\right) KPa$

Os valores dos parâmetros pertinentes ao problema são dados a seguir:

$H = 5\ m$
$E = 210\ GPa$
$\rho = 7800\ Kg/m^3$
$w = 2\ KN/m$
$P = 4\ KN$
$I = \frac{\pi}{64}(d_o^4 - d_i^4)$
$M = (PH + 0,5wH^2)\ KN \cdot m$
$S = (P + wH)\ KN$

Os limites laterais das variáveis de decisão são:

$5 \leq d_o \leq 50\ cm$
$4 \leq d_i \leq 45\ cm$

Considerações para a resolução: *trata-se de um problema não-linear, de duas variáveis reais, uniobjetivo e com restrições, por isso, o método da penalidade deve ser aplicado ao resolver esse problema por meio do algoritmo da ED.*

Bibliografia

[1] ABN. *Agência Brasileira de Notícias, SUSE Linux é usado por mais de 1/3 dos 100 melhores supercomputadores do mundo.* Online. Acessado em 03/09/2014, `http://www.abn.com.br/editorias1.php?id=73841`.

[2] ABRAHAM, A.; JAIN, L.; GOLDBERG, R. *Evolutionary Multiobjective Optimization Theoretical Advances and Applications*. 1. ed. [S.l.]: Springer London, 2005. ISBN 978-1-84628-137-2.

[3] ANDERSON, B. D. O.; MOORE, J. B. *Optimal Control: Linear Quadratic Methods*. [S.l.]: Dover Publications, 2007. ISBN 9780486457666.

[4] ANTONIOU, A.; LU, W.-S. *Practical Optimization Algorithms and Engineering Applications*. 1. ed. [S.l.]: Springer New York, NY, 2007. ISBN 978-0-387-71107-2.

[5] ARORA, J. S. *Introduction to Optimum Design*. Singapore, Roma: McGraw-Hill Book Company, 1987.

[6] BAILI, M. *Analise et Classification de Manipulateur 3R à axes Orthogonaux*. Tese (Doutorado) — University of Nantes, France, 2004.

[7] BAZARRA, M. S.; JARVIS, J. J.; SHERALI, H. D. *Linear Programming and Network Flows*. 4. ed. [S.l.]: Wiley, 2009. ISBN 0470462728.

[8] BAZARRA, M. S.; SHERALI, H. D.; SHETTY, C. M. *Nonlinear Programming: Theory and Algorithms*. [S.l.]: Wiley-Interscience, 2006. ISBN 0471486000.

[9] BIRGE, J. R.; LOUVEAUX, F. *Introduction to Stochastic Programming*. 3. ed. [S.l.]: Springer New York, NY, 2011. ISBN 978-1-4614-0237-4.

[10] BOYD, S.; VANDENBERGHE, L. *Convex Optimization*. [S.l.]: Cambridge University Press, 2004. ISBN 0 521 83378 7.

[11] BRANDÃO, M. A. L. et al. A comparative study using shuffled complex evolution and the differential evolution applied to robotic manipulator design. *10th World Congress on Computational Mechanics, Computational Mechanics 2012 Proceedings*, 2012.

[12] BRANDÃO, M. A. L.; SARAMAGO, S. d. F. P. *Métodos estocásticos de otimização: Algoritmos Genéticos e Evolução Diferencial*. São Carlos, SP: SBMAC, 2011. ISBN 978-85-86883-51-4.

[13] BRANDÃO, M. A. L.; SARAMAGO, S. F. P.; DORICIO, J. L. Optimum desing of 3r robot manipulator by using improved differential evolution implemented in parallel computation. *Brazilian Electronic Journal of Mathematics*, v. 1, n. 2, p. 83–103, 2020.

[14] BRANKE, J. et al. *Multiobjective Optimization Interactive and Evolutionary Approaches*. 1. ed. [S.l.]: Springer Berlin, Heidelberg, 2008. ISBN 978-3-540-88908-3.

[15] BUEHREN, M. *Differential Evolution. MATLAB Central File Exchange, 2023.* Acessado em 18/08/2023, `https://www.mathworks.com/matlabcentral/fileexchange/18593-differential-evolution`.

[16] BUENO, A. D. *Introdução ao Processamento Paralelo e ao Uso de Clusters de Workstacions em Sistemas GNU/LINUX Parte I: Filosofia.* Online. Acessado em 12/11/2002, `http://www.inf.ufes.br/~luciac/fem/117-IntroducaoProgramacaoParalelaECluster-P1.pdf`.

[17] COELHO, C. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2. ed. [S.l.]: Springer, 2007. ISBN 978-0387332543.

[18] COLLETTE, Y.; SIARRY, P. *Multiobjective Optimization Principles and Case Studies*. 1. ed. [S.l.]: Springer Berlin, Heidelberg, 2013. ISBN 978-3-662-08883-8.

[19] COMPANY, O.; PIERROT, F.; FAUROUX, J. C. A method for modeling analytical stiffness of a lower mobility parallel manipulator. *Proc. Of IEEE ICRA: Int. Conf. On Robotics and Automation, Barcelona, Spain*, 2005.

[20] DEBLAISE, D.; HERNOT, X.; MAURINE, P. A systematic analytical method for pkm stiffness matrix calculation. *IEEE International Conference on Robotics and Automation*, 2006.

[21] DEKKER, T. J.; HOFFMANN, W.; POTMA, K. Parallel algorithms for solving large linear systems. *Journal of Computational and Applied Mathematics*, n. 50, p. 221–232, 1994.

[22] EL-KHASAWNEH, B. S.; FERREIRA, P. M. Computation of stiffness and stiffness bounds for parallel link manipulator. *J. Machine Tools & manufacture*, n. v. 39(2), p. 321–342, 1999.

[23] FLOUDAS, C. A. *Deterministic Global Optimization Theory, Methods and Applications*. [S.l.]: Springer New York, NY, 2013. ISBN 978-1-4757-4949-6.

[24] GILL, P. E.; MURRAY, W.; WRIGHT, M. H. *Practical Optimization*. [S.l.]: Academic Press, 1981. ISBN 0122839528.

[25] GITHUB, 2023. Acessado em 08/08/2023, `https://github.com/aaugustoag/Evolucao_Diferencial_AG`.

[26] HILLIER, F. S.; LIEBERMAN, G. J. *Introduction to Operations Research*. 10. ed. [S.l.]: McGraw-Hill Education, 2014. ISBN 0073523453.

[27] HORST, R.; TUY, H. *Global Optimization Deterministic Approaches*. [S.l.]: Springer New York, NY, 2013. ISBN 978-3-662-03199-5.

[28] HSIEH, J. *High-Performance Computing with Beowulf Clusters. p. 75-79, June, 2000*. Online. Acessado em 27/08/2011, `http://www.dell.com/powersolutions`.

[29] HWANG, K. *Advanced Computer Architecture: Parallelism, Scalability, Programmability*. [S.l.]: McGraw-Hill, 1993. ISBN 0-07-031622-8.

[30] KALL, P.; MAYER, J. *Stochastic Linear Programming: Models, Theory, and Computation*. 1. ed. [S.l.]: Springer US, 2006. ISBN 978-0387233857.

[31] LEON, S. J. *Algebra Linear: com aplicações*. 8ª. ed. Rio de Janeiro: LTC, 2011.

[32] MACENA, G. *Github, 2023*. Acessado em 08/08/2023, `https://github.com/Gabriel-Macena/evolucao-diferencial`.

[33] NAIDU, D. S. *Optimal Control Systems*. [S.l.]: CRC Press, 2002. ISBN 0849308925.

[34] NEMHAUSER, G.; WOLSEY, L. *Integer and Combinatorial Optimization*. [S.l.]: John Wiley & Sons, Inc., 1998. ISBN 9780471828198.

[35] NOCEDAL, J.; WRIGHT, S. *Numerical Optimization*. [S.l.: s.n.], 2000. (Springer Series in Operations Research).

[36] OLIVEIRA, G. *Projeto ótimo de robôs manipuladores 3r considerando a topologia do espaço de trabalho*. Tese (Doutorado) — Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, 2011.

[37] OLIVEIRA, G. et al. Projeto Ótimo de manipuladores 3r ortogonais considerando a sua topologia. *Proceedings of the 5th Congresso nacional de engenharia Mecânica - CONEM, Associação Brasileira de Mecânica Computacional - ABMEC, Salvador, Brasil*, 2008.

[38] OLIVEIRA, G. et al. aximização do espaço de trabalho de manipuladores 3r usando algoritmos evolutivos. *XXVII Iberian Latin-American Congress on Computational Methods in Engineering (CILAMCE), Belém do Pará, Brasil. Anais do XXVII CILAMCE (CD-ROM). Associação Brasileira de Mecânica Computacional - ABMEC*, 2006.

[39] OLIVEIRA, L. S.; SARAMAGO, S. F. P. Multiobjective optimization techniques applied to engineering problems. *Journal of the Brazilian Society of Mechanical Sciences and Engineering (Impresso)*, n. v.XXXII, p. 94 – 104, 2010.

[40] OPTIMA. *Global optimization test problems*. Acessado em 21/08/2014, `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm`.

[41] PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity*. [S.l.]: Dover Publications, 1998. ISBN 0486402584.

[42] PARKER, R. G.; RARDIN, R. L. *Discrete Optimization*. [S.l.]: Academic Press, 1988. ISBN 0125450753.

[43] PAUL, R. P.; STEVENSON, C. N. Kinematics of robot wrists. *The Int. J. Robotics Res. 2*, n. 1, p. 1983, 31–38.

[44] PURCINA, L. A. *Técnicas de Otimização Evolutivas aplicadas à Solução de Grandes Sistemas Lineares*. Tese (Doutorado) — Universidade Federal de Uberlândia, 2010.

[45] QUENTAL, N. C. *Modelagem de Desempenho de Programas Paralelos Utilizando Redes de Petri Temporizadas. Trabalho de Conclusão de Curso em Engenharia da Computação pela Escola Politécnica de Pernambuco*. Dissertação (Mestrado) — Universidade de Pernambuco, Pernambuco, 2006.

[46] RITT, M. *Lista de exercícios: Modelagem matemática Otimização Combinatórial*. Online. Acessado em 21/12/2022, `https://www.inf.ufrgs.br/~mrpritt/lib/exe/fetch.php?media=inf05010:lista1_v4.pdf`.

[47] ROCHA, R. *Notas de Aula*. Online. Acessado em 24/06/2014, `http://www.dcc.fc.up.pt/~ricroc/aulas/0708/ppd`.

[48] RUGGIERO, M. A.; LOPES, V. L. R. *Cálculo Numérico - Aspectos Teóricos e Computacionais*. 2ª. ed. São Paulo: Makron Books, 1996.

[49] SAAD, Y.; SCHULTZ, M. H. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, n. 7(3), p. 856–869, 1986.

[50] SAHINIDIS, N. V. Global optimization and constraint satisfaction: The branch-and-reduce approach. in:global optimization and constraint satisfaction first international workshop global constraint optimization and constraint satisfaction, cocos 2002, valbonne-sophia antipolis, france, october 2-4, 2002, revised selected papers. *Global Optimization and Constraint Satisfaction*, n. 1, p. 1–16, 2003.

[51] SARAMAGO, S. F. P.; BERGAMASCHI, P. R.; OLIVEIRA, G. T. S. Optimization of the workspace volume of 3r manipulators using hybrid methodologies. *19th International Congress of Mechanical Engineering, Cobe, ABCM*, n. v.1, p. 1–10, 2007.

[52] STALLINGS, W. *Computer Organization and Architecture Designing for Performance*. 8. ed. [S.l.]: Pearson Prentice Hall, 2010.

[53] STENGEL, R. F. *Optimal Control and Estimation*. [S.l.]: Dover Publications, 2012. ISBN 978-0486682006.

[54] STRANG, G. *Mathematical Methods for Engineers II - Solving Large Linear Systems*. Online. Acessado em 29/08/2014, `http://archive.org/details/flooved1698`.

[55] TAHA, H. A. *Operations Research An Introduction*. [S.l.]: Pearson, 2017. ISBN 1-292-16554-5.

[56] URYASEV, S.; PARDALOS, P. M. *Stochastic Optimization Algorithms and Applications*. 1. ed. [S.l.]: Springer New York, NY, 2013. ISBN 978-1-4757-6594-6.

[57] VANDERPLAATS, G. *Numerical Optimization Techniques for Engineering Design*. USA: McGraw-Hill, 1984.

[58] VINOGRADOV, I. B. et al. Details of kinematics of manipulators with the method of volumes (in russian). *Mekhanika Mashin*, n. 27-28, p. 5–16, 1971.

[59] YANG, D. C.; LAI, Z. C. On the conditioning of robotic manipulators-service angle. *ASME J. Mechanisms, Transm., and Auto*, n. in Design 107, p. 262–270, 1985.

[60] YOSHIKAWA, T. Manipulability of robotic mechanisms. *The Int. J. Robotics Res. 4*, n. 2, p. 3–9, 1985.

Índice