**UNIVERSIDADE FEDERAL DE ALAGOAS-UFAL**
**INSTITUTO DE COMPUTAÇÃO-IC**
**MESTRADO EM MODELAGEM COMPUTACIONAL DE CONHECIMENTO**

**MAX SANTANA ROLEMBERG FARIAS**

# ALGORITMOS EVOLUCIONÁRIOS APLICADOS AO PROBLEMA DO CAIXEIRO VIAJANTE MULTIOBJETIVO

**MACEIÓ**
**2008**

**MAX SANTANA ROLEMBERG FARIAS**

# ALGORITMOS EVOLUCIONÁRIOS APLICADOS AO PROBLEMA DO CAIXEIRO VIAJANTE MULTIOBJETIVO

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional de Conhecimento da Universidade Federal de Alagoas como requisito para obtenção do título de Mestre em Modelagem Computacional de Conhecimento.

Orientador: Prof. Dr. Henrique Pacca L. Luna

Co-orientador: Prof. Dr. Marco César Goldbarg

**MACEIÓ**

2008

**Catalogação na fonte**
**Universidade Federal de Alagoas**
**Biblioteca Central**
**Divisão de Tratamento Técnico**
**Bibliotecária Responsável: Helena Cristina Pimentel do Vale**

F244a Farias, Max Santana Rolemberg.
Algoritmos evolucionários aplicados ao problema do caixeiro viajante multiobjetivo / Max Santana Rolemberg Farias. – Maceió, 2008.
107 f. : il.

Orientador: Henrique Pacca L. Luna.
Co-Orientador: Marco César Goldbarg.
Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2008.

Bibliografia: f. 96-107.

1. Algoritmos evolucionários. 2. Otimização combinatória – Multiobjetivo. 4. Problema do caixeiro viajante. I. Título.

CDU: 004.421

**UNIVERSIDADE FEDERAL DE ALAGOAS-UFAL**
**INSTITUTO DE COMPUTAÇÃO-IC**
**MESTRADO EM MODELAGEM COMPUTACIONAL DE CONHECIMENTO**

# MAX SANTANA ROLEMBERG FARIAS

# ALGORITMOS EVOLUCIONÁRIOS APLICADOS AO PROBLEMA DO CAIXEIRO VIAJANTE MULTIOBJETIVO

Dissertação de Mestrado apresentada ao Instituto de Computação-IC da Universidade Federal de Alagoas, como requisito final para a obtenção do grau de Mestre em Modelagem Computacional de Conhecimento.

**Aprovada em 14 de março de 2008.**

**BANCA EXAMINADORA**

Henrique Pacca Loureiro Luna
Doutor em Automática Otimização – Universite de Toulouse III (Paul Sabatier)/França

Marco Cesar Goldbarg
Doutor em Engenharia de Sistemas e Computação – Universidade Federal do Rio de Janeiro/Brasil

Reinaldo Morabito Neto
Doutor em Engenharia de Transportes – Universidade Federal de São Paulo/Brasil

João Inácio Soletti
Doutor em Engenharia Química – Universidade Federal do Rio de Janeiro/Brasil

É com muito orgulho que
dedico este trabalho à
minha mãe, ao meu pai,
aos meus irmãos e à
minha esposa, pois tenho
certeza de que mesmo
distantes estamos sempre
unidos pelo amor.

# AGRADECIMENTOS

Aos Professores Dr. Henrique Pacca L. Luna e Dr. Marco César Goldbarg, orientadores, pela sua dedicação e pelo seu apoio indispensáveis à realização deste trabalho de pesquisa;

A Andréa, companheira de bons momentos;

Aos amigos Jonildo, Augusto, Marco, Welson e Gustavo, da Secretaria de Tecnologia da Informação da Fundação Universidade Federal do Vale do São Francisco, pela colaboração durante o mestrado;

Aos colegas e alunos do Colegiado de Engenharia da Computação da Fundação Universidade Federal do Vale do São Francisco, pela colaboração.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

Este trabalho apresenta uma visão geral sobre os principais conceitos da otimização combinatória multiobjetivo, onde apresentamos as técnicas mais utilizadas para a resolução de problemas desta natureza. Ao falarmos das técnicas, discutiremos também aspectos importantes quanto aos parâmetros envolvidos em cada técnica, mostrando as principais abordagens utilizadas. Inicialmente, implementamos e testamos o *Multiple Objective Genetic Algorithm* (MOGA) para gerar um conjunto de soluções dominantes próximo ao conjunto de Pareto ótimo para o problema do caixeiro viajante biobjetivo. Em uma segunda fase, implementamos o *Strength Pareto Evolutionary Algorithm* (SPEA) aplicado ao caixeiro viajante biobjetivo.

**Palavras-chave:** Algoritmos evolucionários, Multiobjetivo, Otimização e Caixeiro viajante.

# ABSTRACT

This work presents a general vision about the main concepts of combinatorial multi-objective optimization, where we present the more used technique for the resolution of problems of this nature. To the speech of the techniques we will also argue important aspects how much to the involved parameters in each technique, swing the main used boardings. Initially we implement and test the Multiple Objective Genetic Algorithm – MOGA to generate a set of dominant solutions near to the Pareto optimal set for the bi-objective Traveling Salesman Problems. In a second phase, we will go to implement the *Strength Pareto Evolutionary Algorithm* (SPEA) applied to bi-objective Traveling Salesman Problems.

**Keywords:** Evolutionary algorithms, Multiple Objective, Optmization, Traveling Salesman.

# CAPÍTULO 1

# INTRODUÇÃO

A otimização combinatória é uma disciplina de tomada de decisões, no caso de problemas discretos, que pode ser encontrada em diversas áreas, tais como problemas de planejamento e programação (*scheduling*) da produção, problemas de corte e empacotamento, roteamento de veículos, redes de telecomunicação, sistemas de distribuição de energia elétrica, problemas de localização, dentre outras.

No início das pesquisas nessa área, os modelos eram restritos a problemas com um único objetivo, onde a solução ótima era obtida por meio da maximização ou minimização de uma função objetivo, com as variáveis de decisão sujeitas às restrições. Mas, a partir do começo de 1970, os modelos e as técnicas associadas evoluíram no sentido de contemplarem um maior número de problemas reais, que não podiam ser solucionados por um único objetivo. Isso porque não é trivial agrupar objetivos diferentes, que utilizam métricas diferentes, em uma única função, pois problemas desse tipo geralmente têm critérios (funções objetivos) conflitantes entre si. Objetivos conflitantes é a regra e não a exceção em diversos problemas reais, e a otimização multiobjetivo é utilizada para tratar essa situação.

Na otimização multiobjetivo, ao contrário da otimização monoobjetivo, em geral, não existem soluções ótimas no sentido de minimizar (ou maximizar) individualmente todos os objetivos. A característica principal de otimização multiobjetivo é a existência de um grande conjunto de soluções aceitáveis, que são superiores às demais. Essas soluções aceitáveis são denominadas soluções Pareto ótimo, ou eficiente. Como existe mais de uma solução para os

problemas de otimização multiobjetivo, então fica a critério do tomador de decisão escolher a solução mais atrativa, visto que atende as restrições do problema.

A escolha do método de resolução a ser utilizado, na otimização de um problema, depende, principalmente, da razão entre a qualidade da solução gerada e o tempo gasto, pelo método, para encontrar a solução. Como a maioria dos problemas são intratáveis, isto é, são problemas para os quais é improvável encontrar uma solução em tempo polinomial, ou seja, não é possível desenvolver um algoritmo exato para resolvê-lo em um tempo razoável, para solucionar problemas desse tipo, é preciso utilizar métodos heurísticos. Esses métodos, quando bem desenvolvidos e adaptados aos problemas, são capazes de apresentar soluções de boa qualidade em tempo compatível com a necessidade presente nos problemas.

Com o desenvolvimento e, principalmente, o sucesso dos métodos heurísticos, em especial as metaheurísicas, os pesquisadores, na década de 1990, interessassem-se pelo método, para aplicá-lo em problemas de otimização combinatória multiobjetivo, considerados difíceis computacionalmente (Ehrgott e Gandibleux, 2000).

Atualmente, as metaheurísticas têm sido aplicadas com muito sucesso, na resolução de problemas multiobjetivos, para gerar um conjunto de soluções Pareto ótimo. Recentemente, muitos pesquisadores vêm propondo extensões de metaheurísticas para resolver problemas multiobjetivos, por exemplo, Coello (2001), Deb (2001), Ehrgot (2000), Jaskiewicz (2002) e Zitzler (1998). Pois as metaheurísstica podem ser implementadas com muita flexibilidade para solucionar problemas de otimização multiobjetivo.

Alguns pesquisadores também vêm propondo uma estratégia básica de metaheuristicas diferentes, mesclando características de busca dos Algoritmos Genéticos com técnicas de busca local.

Apesar de toda essa evolução, as técnicas de otimização multiobjetivo atuais ainda não estão fechadas e definidas. Assim, ainda, podem ser propostos muitos algoritmos, e também podem ser desenvolvidas outras técnicas para a avaliação e comparação de resultados.

O objetivo principal deste trabalho é estudar e apresentar algumas técnicas para resolver tipos clássicos de problemas multiobjetivos. Estamos

interessados, especificamente, em problemas multiobjetivos, que envolvem a localização de inúmeras soluções que satisfazem alguns critérios e restrições. São os chamados problemas de Otimização Combinatória Multiobjetivo.

Neste trabalho, iremos oferecer uma visão geral sobre os principais conceitos envolvidos nesta área da pesquisa operacional, apresentando as técnicas mais utilizadas para a resolução de problemas dessa natureza. Ao fazermos isso, discutiremos aspectos importantes quanto aos parâmetros envolvidos em cada algoritmo, mostrando as abordagens utilizadas. Depois de apresentada toda a teoria necessária para o bom entendimento do assunto, será apresentada, também, uma descrição detalhada das metaheurísticas mais comum na literatura especializada. Tomaremos o tradicional Problema do Caixeiro Viajante (*Traveling Salesman Problem*) como um exemplo comum, para mostrar um problema prático de otimização multiobjetivo.

Este trabalho está dividido em 8 capítulos com os seguintes conteúdos. No capítulo 1 é apresentada uma breve introdução sobre a otimização multiobjetivo. No capítulo 2 são apresentados os conceitos básicos utilizados em otimização multiobjetivo. No capítulo 3 são apresentadas as técnicas de solução clássicas para solucionar problemas de otimização multiobjetivo. No capítulo 4 são descritos os algoritmos evolucionários clássicos com multiobjetivo da literatura. No capítulo 5 descreve o problema do caixeiro viajante (PCV) tradicional e apresenta um histórico sobre o problema. No capítulo 6 descreve o problema do caixeiro viajante multiobjetivo, apresentando técnicas de solução e aplicação prática. No capítulo 7 são apresentados os resultados de dois algoritmos evolucionários com multiobjetivo aplicado ao problema do caixeiro viajante multiobjetivo. No capítulo 8, apresentam-se as conclusões do trabalho e os trabalhos futuros.

# CAPÍTULO 2

# OTIMIZAÇÃO MULTIOBJETIVO

Neste capítulo, apresentaremos a formulação matemática de um problema de otimização multiobjetivo e os conceitos básicos usados nesse tipo de problema. Serão descritos, também, algumas técnicas de solução.

## 2.1. Problema de Otimização Multiobjetivo

Um Problema de Otimização Multiobjetivo (POMO) trabalha com mais de uma função objetivo, simultaneamente, buscando-se a otimização do conjunto das funções objetivo por meio de critérios e julgamento das alternativas de solução do problema. Com isso é possível contemplarmos um maior número de problemas reais, de tomada de decisão, que não podem ser solucionados pelas técnicas tradicionais de otimização. Isso porque não é trivial agrupar, em uma única função objetivo, objetivos (critérios) que, na maioria das vezes, são conflitantes entre si.

Um exemplo de um problema com objetivos conflitantes é a tarefa de comprar um computador. A aquisição ótima é um equipamento com custo mínimo e desempenho máximo. Esses objetivos são conflitantes entre si, já que existirão desde computadores com elevado custo e desempenho até aqueles com baixo custo e desempenho.

A Figura 1 mostra a representação gráfica das funções objetivo (preço e desempenho) do problema citado, onde podemos observar, no espaço de objetivos, as soluções que superam outras, também chamadas de soluções não-dominadas, e as que são superadas por, pelo menos, uma outra solução, as chamadas soluções dominadas.

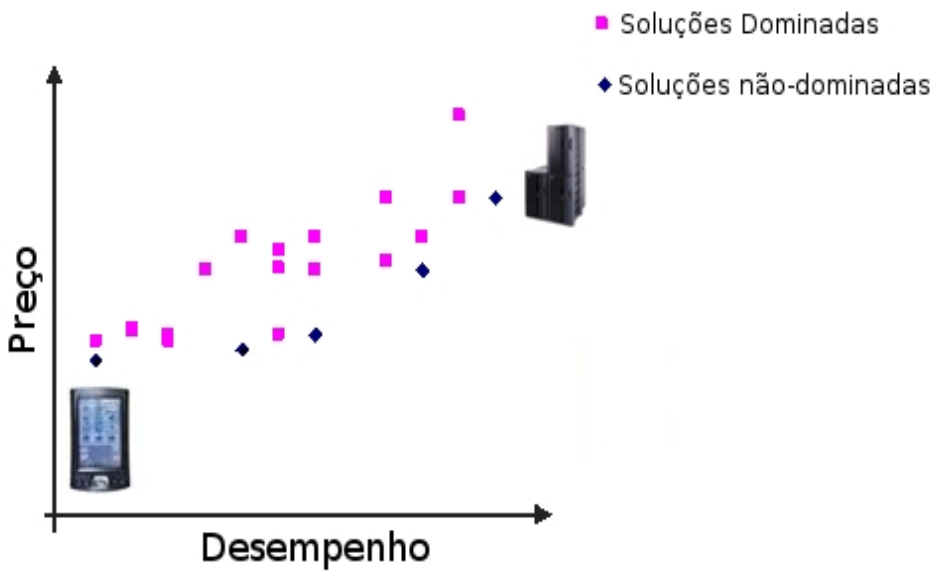



Figura 1: Espaço dos Objetivos.

Através da Figura 1, observamos que nenhuma solução que tenha menor custo e desempenho pode ser considerada superior a outra com maior custo e desempenho. Entretanto, dentre todas as soluções, existem algumas que são superiores a outras, ou seja, apresentam desempenho maior ou equivalente por um custo menor ou igual. Essas soluções, que superam as outras, são as chamadas soluções não-dominadas, enquanto as soluções que são superadas por, pelo menos, uma outra solução são chamadas de soluções dominadas.

### 2.1.1. Formulação

Um POMO pode ser definido formalmente como um processo de otimização, onde se deseja encontrar um vetor $x = [x_1, x_2, ..., x_n]^T \in V$ que satisfaça às restrições do problema. O enunciado geral para o POMO é o seguinte (Deb, 2001):

$$
\begin{array}{rll}
\textit{maximizar/minimizar} & f_m(x), & q = 1, 2, ..., Q \\
\textit{sujeito a:} & g_j(x) \geq 0, & j = 1, 2, ..., J \\
 & h_k(x) = 0, & k = 1, 2, ... K \\
 & x_i^L \leq x_i \leq x_i^U, & i = 1, 2, ..., n
\end{array}
$$

O vetor $x = [x_1, x_2, ..., x_n]^T \in V$, representa o vetor de $n$ variáveis de decisão. Se essas variáveis forem discretas, o problema de otimização multiobjetivo será chamado de problema de otimização combinatória multiobjetivo (Jaszkiewicz, 2001). Esse vetor pertencente a uma região do espaço $\mathfrak{R}^m$ chamada região de visibilidade, ou espaço de objetivos, de $V$. O vetor $x$ também será referido como solução.

Os valores $x_i^L$ e $x_i^U$ representam para as variáveis $x_i$ o mínimo e máximo valor respectivamente. Esses limites definem o espaço de variáveis de decisão ou espaço de decisão $D$.

As $J$ desigualdades $(g_j(x) \geq 0)$ e as $K$ igualdades $(h_k(x) = 0)$ são chamadas de funções de restrições. Uma solução $x$ factível será aquela que atende as $J + K$ funções de restrições impostas por $g_j(x)$ e $h_k(x)$ e os $2n$ limites. O conjunto das soluções factíveis forma o espaço de busca $S$ ou a região factível (Deb, 2001). Na Figura 2, podemos observar o espaço de soluções factíveis de um problema de minimização com dois objetivos.

Para que seja possível otimizar um POMO, é necessário converter todas as $M$ funções objetivo $(f_m(x))$ para maximizar ou minimizar.

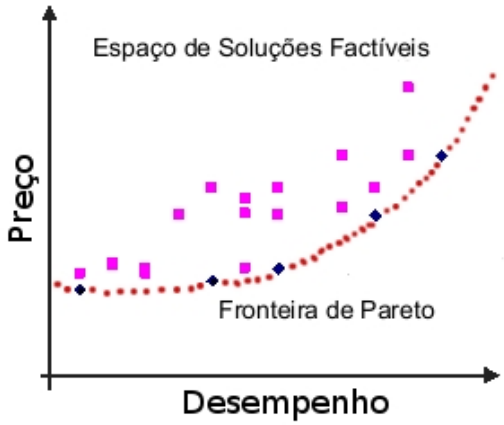



Figura 2: Espaço de Soluções Factível.

## 2.2. Solução Ideal

Solução ideal, ou utópica, também chamada de vetor ideal, representa um vetor *x* que consegue achar os valores ótimos para o problema, ou seja, o máximo ou o mínimo, para cada uma das *M* funções objetivo *($f_m(x)$)*.

Em raríssimas situações, é que é possível encontrar uma solução ideal. Daí chamá-la de solução utópica.

## 2.3. Dominância de Pareto

O conceito de dominância foi introduzido por Vilfredo Pareto, no século XIX, representando, assim, o inicio das pesquisas em otimização multiobjetivo. A dominância de Pareto é utilizada para fazer uma comparação entre duas soluções de um POMO.

Em um POMO, o espaço de objetivos, geralmente, não é completamente ordenado, como no espaço de objetivos de um problema de otimização de um único objetivo, mas é parcialmente ordenado (Pareto, 1896). Essa ordenação parcial é responsável pela distinção básica entre problemas de otimização multiobjetivo.

Assim, podemos descrever as soluções ótimas de Pareto para um POMO, pela seguinte definição de dominância (Deb, 2001):

**Definição 1**: *Uma solução $x_1$ domina outra solução $x_2$ ($x_1 \preceq x_2$) se as seguintes condições forem satisfeitas:*

1. A solução $x_1$ não é pior que $x_2$ em todos os objetivos, ou seja, $f_m(x_1)$ $f_m(x_2)$ para todo $m = 1, 2, ..., M$.
2. A solução $x_1$ é estritamente melhor que $x_2$ em pelo menos um objetivo, ou seja, $f_m(x_1) \lhd f_m(x_2)$ pelo menos para um $m = 1, 2, ..., M$.

O operador $\lhd$ entre duas soluções ($x \lhd y$), significa que a solução x é melhor que y em um objetivo em particular. Reciprocamente, $x \rhd y$ denota que a solução *x* é pior que *y* para algum objetivo.

Em outras palavras, podemos dizer que uma solução $x_1$ domina $x_2$ se e somente se $f_m(x_1) \geq f_m(x_2)$ para todo $m = 1, 2, ..., M$ e $\exists_m \rightarrow f_m(x_1) > f_m(x_2)$, tratando-se de um problema de maximização.

Se as condições são satisfeitas, podemos dizer que a solução $x_2$ é dominada por $x_1$ e $x_1$ é não-dominada por $x_2$ e, também, que $x_1$ é não inferior a $x_2$.

Para demonstrar o conceito de dominância de Pareto, em um POMO, vamos usar o exemplo, já citado, que descreve a tarefa de comprar um computador. A Figura 3 ilustra algumas opções de compra.

O objetivo desse problema é minimizar o preço e maximizar o desempenho. Nesse caso particular, temos cinco alternativas de compra. Por intuição, eliminamos a solução $S_3$, visto que a solução $S_4$ oferece maior desempenho pelo mesmo preço. Pelo mesmo motivo, eliminamos também a solução $S_1$. Utilizando o conceito de dominância de Pareto, podemos dizer que a solução $S_4$ domina a solução $S_3$ ($S_4 \preceq S_3$), a solução $S_2$ domina a solução $S_1$ ($S_2 \preceq S_1$) e as soluções $S_2$, $S_4$ e $S_5$ são boas alternativas de compra, pois são não-dominadas por nenhuma outra. Com isso podemos concluir que o conceito de dominância consegue comparar soluções com múltiplos objetivos.

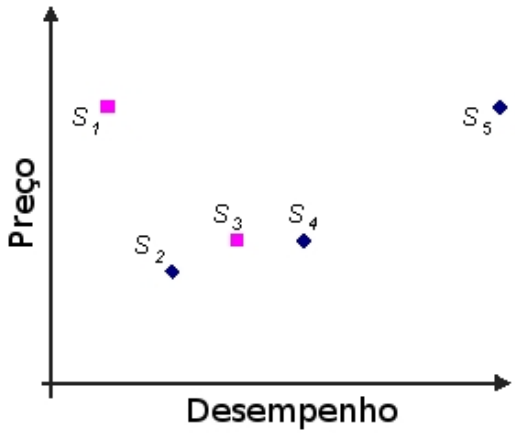


Figura 3: Algumas opções de compra de um computador.

### 2.3.1. Propriedade da Relação de Dominância

A relação de dominância satisfaz as três propriedades a seguir:

1. Não reflexiva. Conforme a definição 1, uma solução não pode ser dominada por ela mesma.
2. Não simétrica. Porque se $S_1 \preceq S_2$ não implica que $S_2 \preceq S_1$.
3. Transitiva. Porque se $S_1 \preceq S_2$ e $S_2 \preceq S_3$ então $S_1 \preceq S_3$.

### 2.3.2. Dominância Fraca e Forte

A relação de dominância pode ser classificada em dominância fraca e forte (Coello, 1998). A definição de dominância forte é semelhante à descrita anteriormente. Já a dominância fraca é definida como:

**Definição 2**: *Uma solução $x_1$ domina fracamente outra solução $x_2$ ($x_1 \prec x_2$) se a seguinte condição for satisfeita:*

1. A solução $x_1$ não é pior que $x_2$ em todos os objetivos, ou seja, $f_m(x_1)$ $f_m(x_2)$ para todo $m = 1, 2, ..., M$.

Em outras palavras, podemos dizer que uma solução $x_1$ domina fracamente $x_2$ se e somente se $f_m(x_1) \geq f_m(x_2)$ para todo $m = 1, 2, ..., M$, tratando-se de um problema de maximização.

Intuitivamente, podemos dizer que uma solução fortemente dominada é, também, fracamente dominada, sendo a recíproca falsa.

## 2.4. Otimalidade de Pareto

Quando o conjunto de soluções viáveis $V$ é finito, é possível comparar duas soluções duas a duas, segundo o conceito de dominância de Pareto. O conjunto $V$ pode ser dividido em dois subconjuntos: conjunto das soluções dominadas e o conjunto das soluções não-dominadas.

### 2.4.1. Pareto Ótimo

**Definição 3**: *Dado o conjunto de soluções V, o conjunto de soluções não-dominadas V' é formado por aquelas soluções que são não-dominadas por qualquer elemento de V*.

Se o conjunto *V* for o espaço completo de busca *(V = S)*, o conjunto *V'* será chamado de conjunto das soluções Pareto ótimo.

Uma solução $x_1$, gerada, será Pareto ótimo se e somente se $x \in V$ e $\neg\exists_j \rightarrow f_j(x) \preceq f(x_1)$. Em outras palavras, podemos dizer que a solução gerada não pode ser dominada por nenhuma outra solução, do espaço de soluções viáveis.

Na Figura 4, podemos ver vários exemplos de conjuntos de Pareto ótimos, conforme várias combinações de objetivos para as funções $f_1$ e $f_2$ .

O conjunto Pareto ótimo pode ser representado de maneira continua, indicando onde o conjunto está localizado, como mostram as curvas na Figura 4.

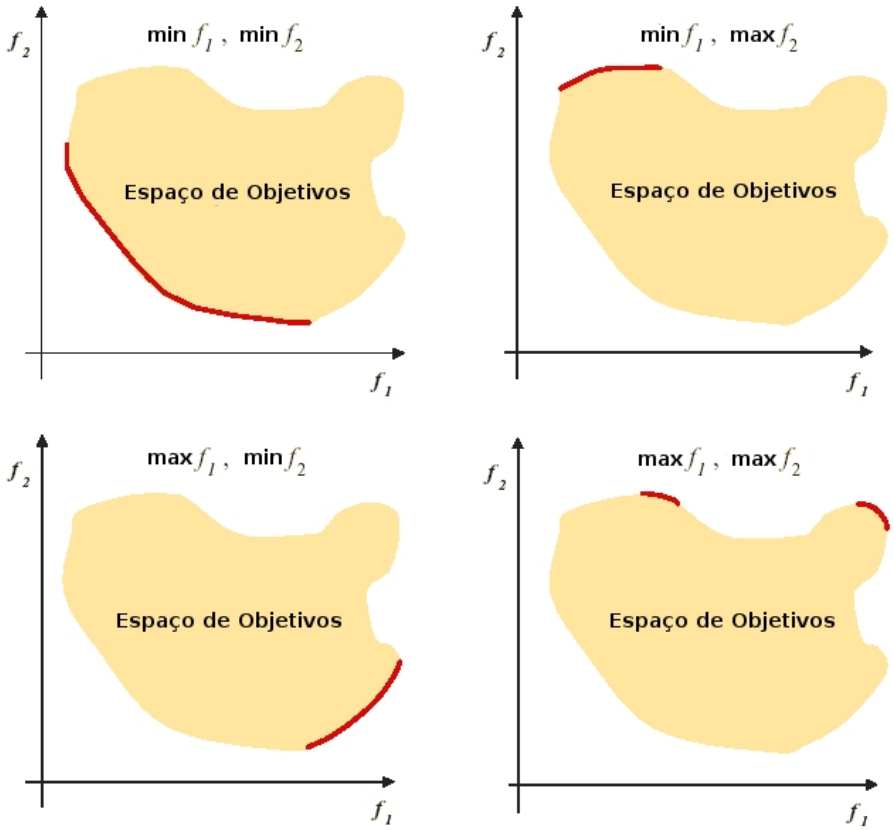



Figura 4: Exemplos de conjuntos Pareto ótimos.

Em um ponto Pareto ótimo, existe, também, a idéia de solução ótima global e localmente ótima, definida como:

**Definição 4**: *O conjunto das soluções não-dominadas para a totalidade do espaço de busca factível S é chamado de conjunto das soluções Pareto ótimo global*.

As soluções contidas nesse conjunto são as soluções ótimas de um POMO.

**Definição 5**: *Se cada elemento x do conjunto P não é dominado por alguma solução y na vizinhança de x tal que* $||y - x||_\infty \geq \varepsilon$, *onde* $\varepsilon$ *é um número positivo arbitrariamente pequeno, então o conjunto P é um conjunto de soluções Pareto ótimo local*.

A Figura 5 mostra um exemplo de uma solução Pareto ótimo local.

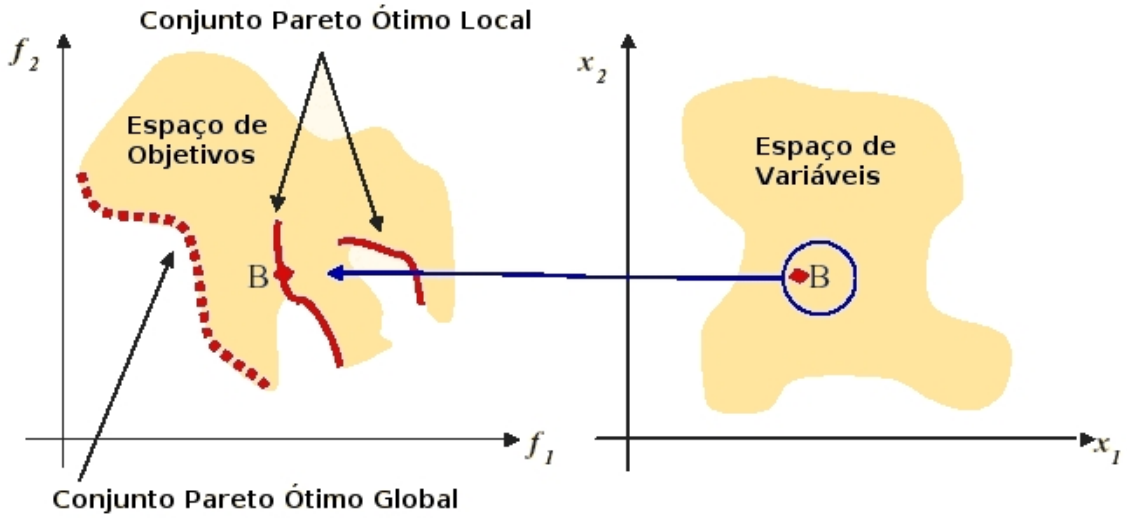



Figura 5: Exemplo de conjunto Pareto ótimo local.

### 2.4.2. Fronteira de Pareto

**Definição 6**: *A fronteira de Pareto está formada pelo conjunto de vetores de funções objetivo* $f(x) = (f_1(x), f_2(x), ..., f_m(x))^T$, *para cada solução x que está no conjunto de Pareto ótimo*.

A fronteira de Pareto, ou curva minimal é uma curva composta com as soluções Pareto ótimo. Essa curva consiste num limite superior para o problema e fica próxima à curva do espaço de objetivos.

Um exemplo de fronteira de Pareto é mostrado na Figura 6, que mostra a curva para o problema da compra de um computador, exemplo citado neste capítulo.

Na maioria dos problemas de otimização multiobjetivo, o objetivo principal é conseguir descrever uma função ou conjuntos de pontos que formem uma curva o mais próximo possível da fronteira de Pareto. Entretanto, para problemas com mais de três dimensões, fica difícil de ilustrar a fronteira de Pareto.

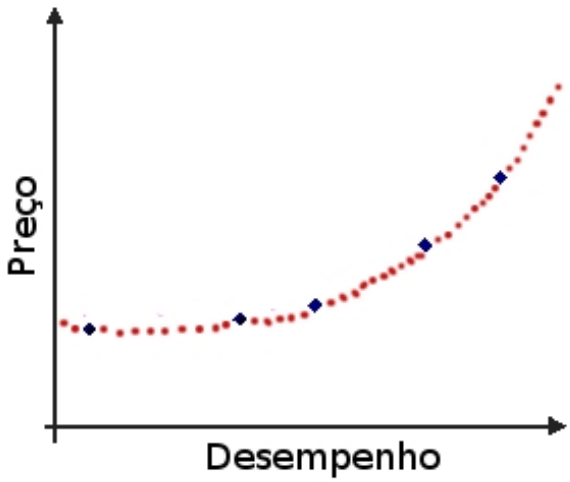



Figura 6: Fronteira de Pareto.

### 2.4.3. Solução Ótima de Pareto

Quando todos os objetivos são simultaneamente considerados, para obter as soluções, estas soluções são chamadas de soluções ótimas de Pareto, como citado anteriormente. A diferença, entre o conjunto de soluções não-dominadas e o conjunto de soluções ótimas de Pareto, é que o conjunto de soluções não-dominadas é definido no contexto de uma amostra do espaço de busca, enquanto o conjunto de soluções ótimas de Pareto é definido em relação a todo o espaço de busca.

## 2.5. Metas em Otimização Multiobjetivo

Duas importantes metas para a otimização multiobjetivo são mostradas por Deb (Deb, 2001):

1. Encontrar um conjunto de soluções o mais próximo possível da fronteira de Pareto.
2. Encontrar um conjunto de soluções com a maior diversidade possível.

A Figura 7 mostra dois gráficos. O primeiro mostra uma boa distribuição de soluções na fronteira de Pareto, enquanto, no outro, as soluções estão distribuídas em apenas algumas regiões da fronteira de Pareto.
Em otimização multiobjetivo, é necessário assegurar a maior cobertura possível da fronteira de Pareto, já que a fronteira representa o conjunto de soluções de interesse no problema multiobjetivo.

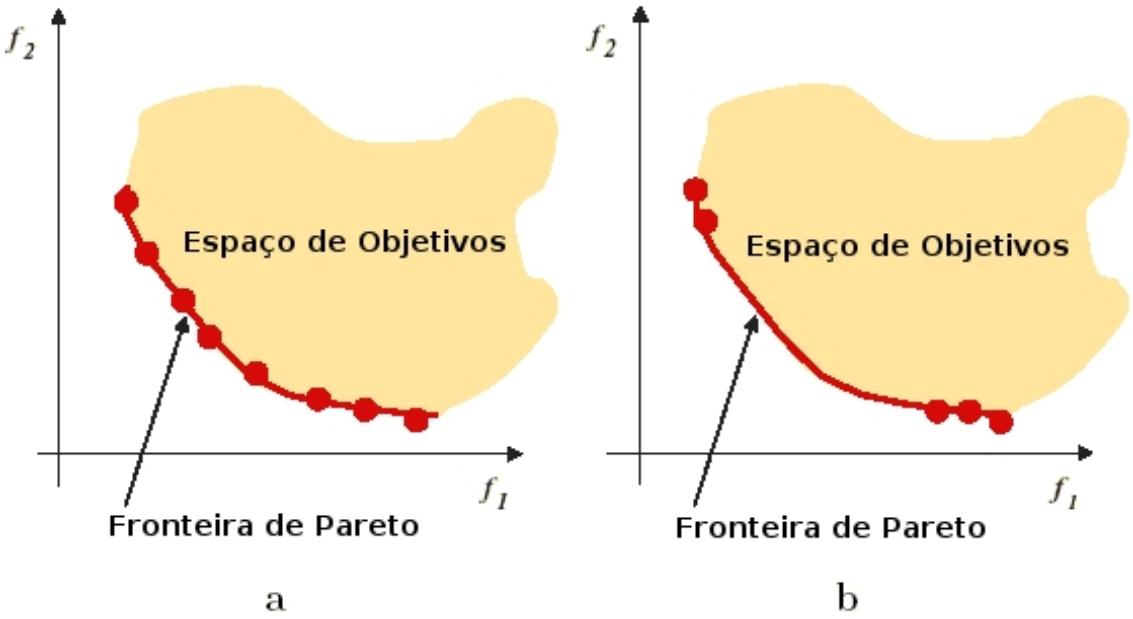



Figura 7: Distribuição de soluções na fronteira de Pareto.

## 2.6. Otimização Monoobjetivo vs Multiobjetivo

Três diferenças importantes entre otimização multiobjetivo e otimização monoobjetivo podem ser identificadas:

1. Em problemas de otimização monoobjetivo, a meta é achar uma solução ótima global, enquanto nos POMO, as metas são: achar o conjunto de soluções da fronteira de Pareto e preservar a diversidade neste conjunto.

2. Um POMO trabalha com o espaço de variáveis e o espaço de objetivos. Já o problema com um único objetivo trabalha focado no espaço de variáveis.

Diferentemente da abordagem de otimização monoobjetivo, que busca otimizar uma função simples, a otimização multiobjetivo busca otimizar o conjunto das funções objetivo, selecionando, assim, a solução de melhor compromisso.

## 2.7. Aplicações

Problemas de telecomunicação é um exemplo de aplicação real e prática de um POMO, pois, ás vezes, é necessário minimizar custo e qualidade (Thiongane et al, 2001).

Outros exemplos de aplicações reais de um problema multiobjetivo são:

- Organização de viagem (Godart, 2001);
- Programação de tripulações de companhias aéreas (Ehrgot e Ryan, 2002);
- Capacidade de infra-estrutura ferroviária (Delorme et al, 2003).

# CAPÍTULO 3

# TÉCNICAS DE SOLUÇÃO MULTIOBJETIVO

As técnicas de solução utilizadas até algum tempo atrás eram simples adaptações, feitas nos algoritmos já existentes para os problemas de otimização com objetivo simples ou, simplesmente, transformando os vários objetivos em um único objetivo.

Mas, hoje, já existem trabalhos, que adotam técnicas específicas para o POMO, utilizando os conceitos de Pareto ótimo e dominância de Pareto.

Um aspecto importante nas técnicas de solução para POMO é a presença do tomador de decisão, que pode ser uma pessoa, ou um grupo, que vai analisar e fazer uma comparação entre vantagens e desvantagens da melhoria de uma função em detrimento de outras. Cabe, também, ao tomador de decisão determinar a relevância de cada função para o problema.

Podemos observar, na Figura 8, a classificação das técnicas, feitas de acordo com o momento no qual o tomador de decisão exerce o seu papel (Andersson, 2000).

## 3.1. Técnicas sem Interferência do Tomador de Decisão

Nesse tipo de técnica, não existe a presença do tomador de decisão. As soluções geradas são a partir de uma fórmula predefinida. Com certeza, essa é a técnica menos utilizada, pelas abordagens propostas. Para esse tipo de técnica, a única abordagem, que se destaca, é a chamada *Min-Max*.

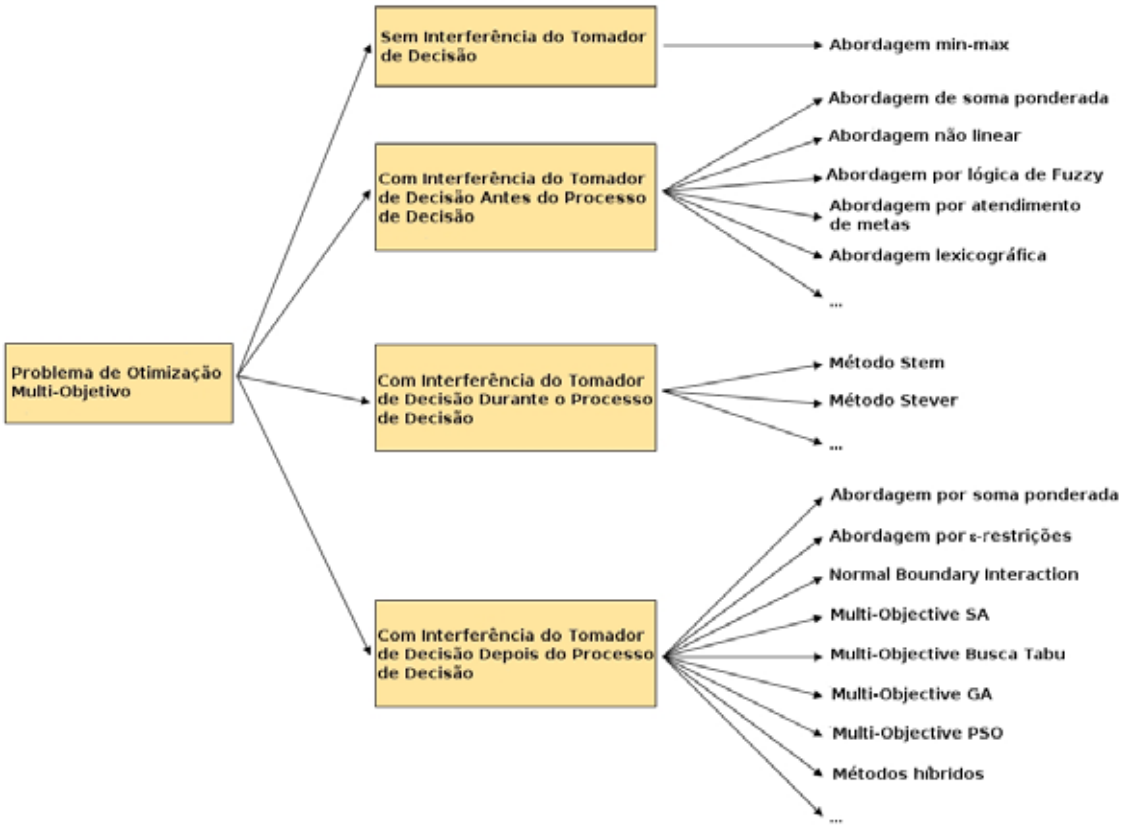



Figura 8: Classificação das técnicas de solução para POMO.

A abordagem *Min-Max* é baseada na minimização da distância relativa entre uma solução candidata e uma solução ideal (utópica), o que pode ser visto na Figura 9. Nesse tipo de abordagem, o POMO é formulado de acordo com a sequação (Andersson, 2000):

$$\min\left[\sum_{i=1}^{k}\left(\frac{f_i(\vec{x})-f^*(\vec{x})}{f^*(\vec{x})}\right)^p\right]^{\frac{1}{p}}$$

O vetor $\vec{x} \in V$ e o expoente *p*, responsáveis pelo cálculo das distâncias, podem receber valores entre um e infinito ($1 \leq p \geq \infty$). Mas, geralmente, os valores usados para *p* são: um, para formulação simples, dois, para distâncias euclidianas, e infinito, para modelos Tchebycheff (Andersson, 2000). Esse tipo de abordagem fornece apenas uma solução na Fronteira de Pareto, que deve ser aceita como solução final.

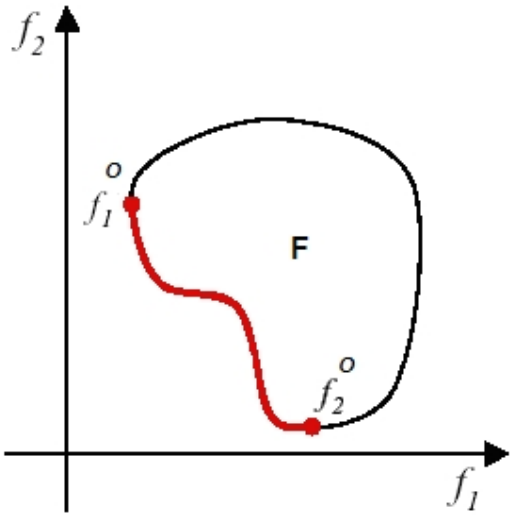


Figura 9: Distância entre duas soluções ideais.

Caso essa abordagem seja usada em uma técnica que tenha a presença do tomador de decisão, será necessário executar o algoritmo várias vezes, com valores diferentes para *p*, com o intuito de obter um conjunto de soluções na Fronteira de Pareto.

## 3.2. Técnicas com Interferência do Tomador de Decisão antes do Processo de Solução

Essa técnica é a mais comum para solucionar POMO. Nesse tipo de técnica, o tomador de decisão informa quais são as suas preferências em relação a cada função objetivo do problema. Essas serão usadas como parâmetros nos algoritmos.

Para utilizar essa técnica, o tomador de decisão deve conhecer os objetivos e as restrições do problema como um todo.

Algumas abordagens foram propostas, utilizando essa técnica, como podemos ver a seguir.

### 3.2.1. Abordagem da Soma Ponderada

Essa abordagem é a mais fácil e talvez a mais utilizada na programação multiobjetivo. A abordagem da soma ponderada tem como objetivo, transformar os vários objetivos do problema em um único objetivo. Tal transformação é feita, usando um “peso” associado a cada função objetivo. Como essa

abordagem não requer nenhuma modificação mais grave na modelagem, podemos aplicá-la a qualquer modelo de aplicação já existente, seja ele linear ou não-linear.

Como as funções objetivo são geralmente de diferentes magnitudes, torna-se necessário normalizá-las primeiro, para que o cálculo da soma dos objetivos não seja distorcido pelos valores de patamares diferentes. Feita a escalarização, é possível formular, assim, uma simples função objetivo, para o problema, conforme a equação abaixo (Andersson, 2000):

$$\min \sum_{i=1}^{k} c_i \, f_i(\vec{x})$$

Onde *c* corresponde ao "peso" associado à função objetivo ($f_i$), esse "peso" irá informar a relativa importância do objetivo para o problema como um todo. O vetor de parâmetros *c* pode assumir quaisquer valores, mas usa-se, normalmente, a seguinte convenção (Andersson, 2000):

$$c \geq 0$$
$$\sum_{i=1}^{k} c_i = 1$$

Como exemplo, podemos observar a Figura 10, onde temos um vetor de pesos $\boldsymbol{c} = (c_1, c_2)$ para cada um dos objetivos. A partir desse vetor **c,** é possível tangenciar o contorno de ***F***, no espaço de objetivos, obtendo-se, assim, uma linha reta que avalia localmente, em termos de função de utilidade do decisor, quanto se pode perder em um critério para se ganhar no outro.

Cada linha de contorno possui o menor valor para ***F***. Para encontrar o menor valor para uma equação, normalizada, do problema é equivalente a achar uma linha de contorno com um valor mínimo para ***F***.

Podemos observar, na Figura 10, várias linhas de contorno para ***F***, sendo que a linha *d* é tangencial a um ponto do espaço de objetivos *(A)*, onde esse ponto se encontra na fronteira de Pareto e, conseqüentemente, é uma solução ótima de Pareto.

Toda solução gerada é analisada pelo tomador de decisão e, caso a solução não seja “boa”, o processo é reinicializado com novos parâmetros (Pesos). Para tentar obter soluções ótimas de Pareto, deve-se usar a abordagem por múltiplas rodadas, que será tratada ainda neste capítulo.

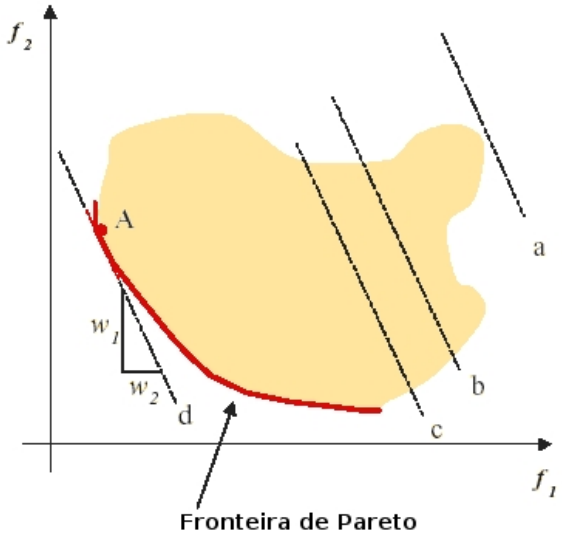


Figura 10: Abordagem da soma ponderada.

### 3.2.2. Abordagem não-linear

Nessa abordagem, as funções objetivo são agrupadas em uma única função objetivo, de acordo com uma equação não-linear. Pelo mesmo motivo da abordagem anterior, essa também deve ser normalizada. Isso pode ser feito dividindo-se o valor de cada função objetivo pelo seu valor da função ideal correspondente.

O real objetivo dessa abordagem é favorecer objetivos que se destacam dos demais, o que é feito elevando-se a solução a um expoente qualquer, e as soluções que se destacarem mais terão vantagens. Um exemplo dessa abordagem foi utilizado por Andersson et. al. (1998), na qual ele utilizou a equação (Andersson, 2000):

$$\min \sum_{i=1}^{k} \left( \frac{f_i(x)}{f_i^{O}(x)} \right)^{p}$$

O expoente *p* representa o fator de diferenciação entre cada unidade das funções objetivo normalizadas. Na Figura 11, podemos observar o efeito não-linear, que contabiliza maiores penalidades para valores maiores do critério, que se quer minimizar (Andersson, 2000).

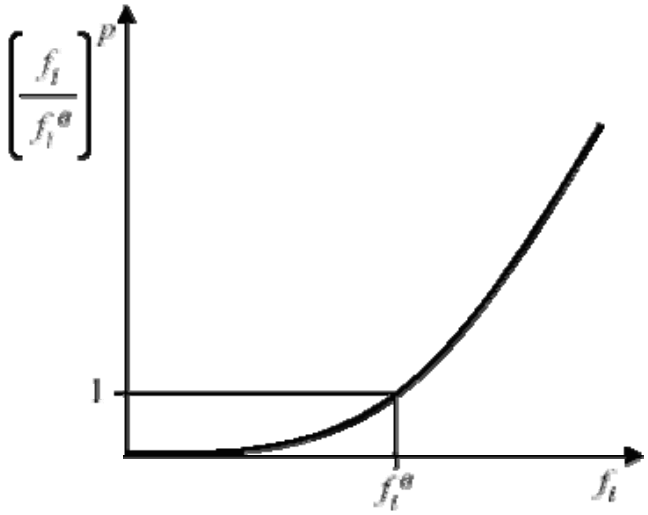



Figura 11: Gráfico de $\left(\frac{f_i}{f_i^o}\right)^p$ por $f_i$ para *p* = 3.

### 3.2.3. Abordagem por lógica Fuzzy

Essa abordagem segue o conceito da Lógica Fuzzy, criada por Lotfi Zadeh (Zadeh, 1994), onde a idéia da Lógica de Fuzzy é mapear valores em atributos, onde, para cada atributo, se tem uma percentagem, que Zadeh chama de grau de pertinência, para classificar o quanto aquele valor pertence a cada atributo. Esse grau é representado pela letra grega $\mu$, onde $\mu$ varia de 0 (não pertence) a 1 (totalmente pertencente).

Na programação multiobjetivo, cada função objetivo terá um valor de normalização associado, que é expresso por (Andersson, 2000):

$$\mu_i(f_i(x))$$

Esse valor expressará o grau de satisfação do objetivo correspondente em relação à solução ideal do problema, onde o tomador de decisão define para cada função qual o comportamento da função de pertinência. Feito isso,

agora, o problema transforma-se em um problema de maximização do grau de pertinência das funções.

Em Chiampi et. al. (1998), são mostradas duas formulações diferentes para a função objetivo de uma abordagem de *fuzzy*, conforme equação abaixo:

$$F_{fuzzy}(x) = \prod_{i=1}^{k} \mu_i(f_i(x))$$

$$F_{fuzzy}(x) = \min(\mu_1(f_1(x)), \mu_2(f_2(x)), ..., \mu_k(f_k(x)))$$

### 3.2.4. Abordagem de Programação de Metas

Essa abordagem foi desenvolvida inicialmente por Charnes et al (1955), onde eles propõem que o tomador de decisão associe a cada função objetivo uma meta. O objetivo dessa abordagem é tentar achar uma solução que possa atingir as metas, mas, caso não exista uma solução factível que alcance as metas, para todos os objetivos, deve minimizar os desvios em relação às metas.

Existem três tipos de critérios para atingir uma meta (Deb, 2001):

1. $f(x) \leq t$, onde o objetivo é minimizar o desvio $p$ para que $f(x) - p \leq t$. Se $f(x) > t$, o desvio $p$ será a quantidade pela qual $f$ supera $t$. Caso contrário, $p$ será zero.
2. $f(x) \geq t$, onde o objetivo é minimizar o desvio n para que $f(x) + n \geq t$. Se $f(x) < t$, o desvio $n$ será a quantidade para $f$ alcance $t$. Caso contrário, $n$ será zero.
3. $f(x) = t$, onde o objetivo é minimizar a soma dos desvios $p$ e $n$ para que $f(x) - p + n = t$. Se $f(x) > t$, $p$ é positivo e $n$ é zero. Se $f(x) < t$, $n$ é positivo e $p$ é zero. Caso $f(x) = t$, ambos os desvios ($p$ e $n$) serão zero.

Os critérios citados acima podem ser resumidos na seguinte restrição genérica:

$$f(x) - p + n = t$$

Cada meta deve ser convertida em uma restrição de igualdade, e, também, devem ser minimizados todos os desvios para poder resolver um problema, usando essa abordagem. Para isso, existem várias formas de trabalhar com essa abordagem.

#### 3.2.4.1. Programação de Metas com Peso

Tal abordagem possui a mesma dificuldade da abordagem de soma ponderada, já citada neste capítulo. Para um problema com *k* objetivos, deve ser formulada uma função com a soma dos desvios para cada um dos *k* objetivos. A forma geral pode ser vista abaixo:

$$\text{minimizar} \quad \sum_{j=1}^{k} (\alpha_j p_j + \beta_j n_j)$$

$$\text{sujeito a} \quad f_j(x) - p_j + n_j = t_j,\ j = 1, 2, \ldots, k$$

$$x \in S$$

$$n_j,\ p_j \geq 0$$

Onde $\alpha_j$ e $\beta_j$ representam os pesos associados aos desvios *p* e *n*, respectivamente, para o *j-ésimo* objetivo. S representa o espaço de decisão factível.

Ao final, se tínhamos *k* objetivos e *r* restrições, teremos agora *r+k* restrições e apenas um objetivo, que seria a minimização da soma dos desvios de cada objetivo.

#### 3.2.4.2. Programação de Metas Lexicográficas

Essa abordagem é utilizada pelo tomador de decisão quando ele pode definir uma ordem de prioridade entre os objetivos a serem otimizados.

Depois que são definidas as prioridades para cada função objetivo $f_i(x)$ tornam-se possível encontrar o conjunto de Pareto ótimo. Mas, para isso, é recomendável antes desprezarmos as funções de menor relevância, transformando-as em uma restrição.

Assim, é possível otimizar a função mais importante, lembrando-nos de que, a cada passo dessa abordagem, devemos tentar otimizar as outras funções de modo a não comprometer os valores das funções já otimizadas.

Essa abordagem é, geralmente, utilizada em combinação com outras. Um exemplo disso é o uso dela com a de programação por metas, no mecanismo de seleção para algoritmos genéticos.

## 3.3. Técnicas com Interferência do Tomador de Decisão durante o Processo de Solução

Essas técnicas também são conhecidas como métodos interativos. Nessas abordagens, o tomador de decisão age simultaneamente com a técnica, e a cada parada do método, ele regula os parâmetros do algoritmo, guiando a solução para um caminho desejado.

A grande vantagem dessas abordagens é que o tomador de decisão age de uma forma mais interativa, informando os parâmetros durante o processo. Essa técnica é um verdadeiro processo de aprendizagem.

Devemos lembrar que o sucesso dessas técnicas está associado à disponibilidade do tomador de decisão, já que ele será muito mais exigido.

### 3.3.1. Método Stem

Esse método foi proposto por Benayoun et al (1971), tendo como estratégia reduzir gradativamente o espaço de busca para o problema o método pode ser chamado de STEM, ou STEP-Method.

O método é formulado através da abordagem *Min-Max*, sem se prender muito aos pesos. As soluções obtidas, em um número *h* de iterações, são orientadas e analisadas pelo tomador de decisão que intervém por meio de respostas precisas às perguntas formuladas pelo algoritmo. Se algumas das soluções apresentarem um valor aceitável, então o problema é reformulado. O procedimento de reformulação consiste em obter um valor aceitável, pelo tomador de decisão, para cada função objetivo.

Esse procedimento será repetido até que o espaço de todas as variáveis do problema atinja valores aceitáveis.

### 3.3.2. Método Steuer

Esse método foi proposto por Steuer e Choo (1983), no qual eles fazem uma amostragem em subconjunto do espaço de soluções não-dominadas,

usando a abordagem de soma ponderada.

O primeiro passo desse método é criar um conjunto de vetores de pesos dos mais diversos. Logo após esses pesos são submetidos à abordagem de soma ponderada, seguindo a norma de *Tchebycheff*.

As soluções encontradas são filtradas, e somente as soluções não-dominantes são mostradas ao tomador de decisão. A partir daí, ele deverá escolher entre elas as melhores. Com base nessa escolha, os valores dos pesos são limitados inferiormente ou superiormente, e, então, executa-se novamente a busca, até o número de iterações predefinidas.

## 3.4. Técnicas com Interferência do Tomador de Decisão depois do Processo de Solução

De todas as técnicas, para a solução POMO, apresentadas, essa é a que vem recebendo a maior atenção ultimamente, pela comunidade cientifica. A grande diferença entre essa técnica e as demais é que, ao invés de gerar uma solução, ela gera um conjunto de soluções, que, no final, será selecionada pelo tomador de decisão, tornando, assim, possível a escolha entre várias soluções.

Podemos destacar, nessa técnica, modificações de diversas metas-heurísticas existentes como *simulated annealing*, busca tabu e algoritmo genético.

### 3.4.1. Abordagem por Múltiplas Rodadas

A maioria dos métodos utilizados para solucionar POMO somente oferece, ao final, uma única solução, considerada a melhor. Mas, a mudança de foco quanto ao resultado de uma única solução para um conjunto de soluções Pareto ótimo, fez com que alguns trabalhos se voltassem para a adaptação dos métodos, para oferecer várias soluções por meio de múltiplas rodadas, fazendo assim modificações de alguns dos parâmetros.

Dessas abordagens, as três que mais se destacam são descritas a seguir.

#### 3.4.1.1. Abordagem por Soma Ponderada

É a abordagem mais simples e direta de se obter um conjunto de pontos, por intermédio de múltiplas rodadas. Para cada conjunto de pesos associados, podemos ter um ponto do conjunto Pareto ótimo a cada execução do algoritmo.

Alguns problemas com essa abordagem foram reportados por Steuer (1986), dentre eles os mais importantes são: a distorção da fronteira de Pareto, que depende dos valores adotados para as execuções, não se analisando, assim, regiões importantes; e a incapacidade de encontrar todos os pontos da fronteira de Pareto, quando o espaço de soluções do problema é não convexo. Na Figura 12, podemos observar, para um POMO de dois objetivos, que nem todos os pontos, Pareto ótimo, admitem linhas de contorno.

Os pontos C e D da figura não possuem linhas de contorno porque esses pontos não podem ser encontrados pela minimização da função *f* do problema.

Para obtermos mais detalhes sobre convexidade em problemas de otimização multiobjetivo, ver Ferreira (1999).

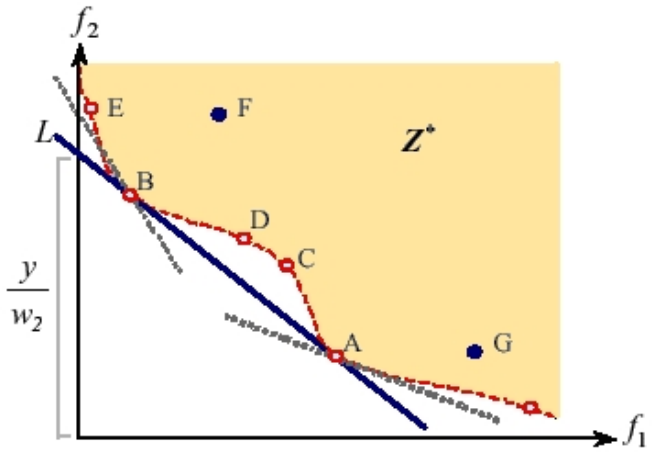



Figura 12: Interpretação gráfica da abordagem por soma ponderada.

Para contornarmos essa limitação, podemos utilizar essa abordagem em junção com a abordagem Min-Max, onde a função objetivo pode ser formulada da seguinte maneira:

$$\min\left[\sum_{i=1}^{k}\left(\lambda_i f_i(\vec{x})\right)^p\right]^{\frac{1}{p}}$$

O valor de $p$ $\left(1 \le p \le \infty\right)$ torna o problema mais difícil quando é aumentado (Andersson, 2000).

#### 3.4.1.2. Abordagem por ε-Restrições

Essa abordagem foi proposta por Haimes et al (1971), e, nela, eles sugeriram reformular um POMO, considerando qualquer objetivo e mantendo como restrições os demais objetivos com valores definidos pelo tomador de decisão. A formulação é definida a seguir (Deb, 2001):

minimizar $f_u(x)$

sujeto a $f_m(x) \le \varepsilon_m$, $m = 1, 2, ..., M$ e $m \neq u$

$g_i(x) \ge 0$, $j = 1, 2, ..., J$

$h_k(x) = 0$, $k = 1, 2, ..., K$

$x_i^{(L)} \le x_i \le x_i^{(U)}$, $i = 1, 2, ..., N$

Onde $\varepsilon_m$ é o valor definido pelo tomador de decisão, e representa um limite máximo para o valor de $f_m$.

Para exemplificar essa abordagem, vamos usar um POMO não convexo de dois objetivos $f_1$ e $f_2$. Vamos escolher $f_2$ e manter $f_1$ como uma restrição ($f_1 \le \varepsilon_1$).

Na Figura 13, podemos observar o espaço de objetivos e os vários valores para $\varepsilon_1$. Podemos observar, também, que o mínimo de $f_2$ vai depender da escolha $\varepsilon$. Por exemplo, se usarmos o $\varepsilon_1^b$, o valor mínimo para $f_2$ é o ponto B. Com isso, podemos concluir que para todos os valores diferentes de $\varepsilon$ achamos diferentes soluções Pareto ótimo. Os resultados dessa abordagem estão garantidos pelo seguinte teorema.

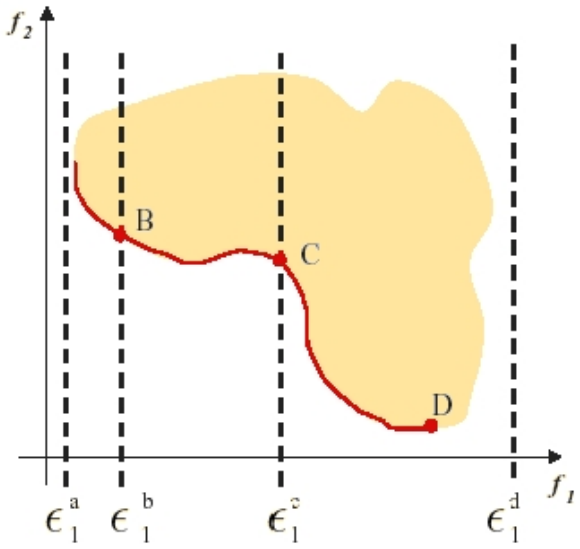


Figura 13: Abordagem por ε-Restrições.

**Teorema 1**: A solução para o problema formulado, pela formulação apresentada, é Pareto ótimo para qualquer vetor $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{u-1}, \varepsilon_{u+1}, \ldots, \varepsilon_M)$.

A Figura 13, também, mostra um exemplo quando ocorre um erro na abordagem. Se o limite não é selecionado adequadamente, por exemplo, o $\varepsilon_1^d$, o subespaço obtido pelas restrições pode ser vazio, isto é, o problema não possui solução. Para evitar essa situação, Cohon (1978) desenvolveu um algoritmo para obter os valores adequados dos limitantes.

#### 3.4.1.3. Normal Boundary Interaction

Tal técnica foi apresentada por Das e Dennis (1998). O objetivo principal dessa técnica, é conseguir descrever a fronteira de Pareto, supondo que os mínimos globais dos objetivos ($f^*$) são conhecidos. Para alcançar esse objetivo, devemos primeiramente estabelecer a envoltória convexa, formada pelos mínimos globais dos objetivos (CHIM, do inglês, Convex Hull of the Individual Mínima). Observe a Figura 14 (Andersson, 2000) para entender melhor o CHIM. Depois que for estabelecido o CHIM, o algoritmo pode encontrar a intersecção entre regiões viáveis do espaço de busca e a envoltória convexa, começando a busca dos pontos espalhados uniformemente pelo CHIN. Essa intersecção será a fronteira de Pareto.

Caso a fronteira de Pareto tenha uma forma muito complexa, a técnica pode encontrar soluções ótimas que não pertençam ao conjunto de Pareto.

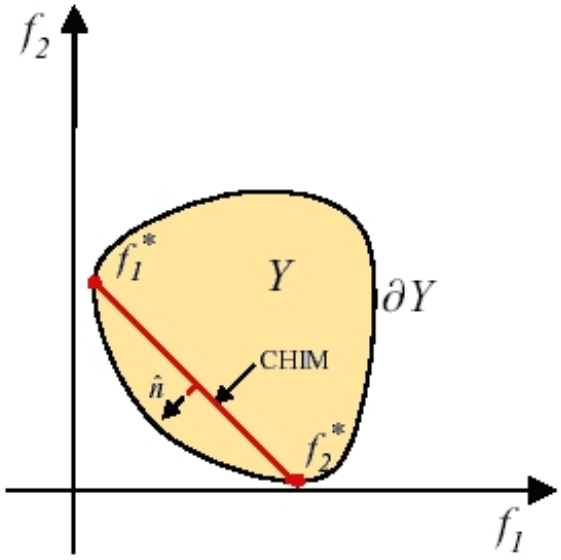



Figura 14: Normal Boundary Interaction.

### 3.4.2. Simulated Annealing

A técnica *Simulated Annealing* (SA), corresponde a uma simulação algorítmica do processo físico de recozimento de certos materiais. Essa técnica foi proposta por Metropolis et al. (1953), Kirkpatrick et al. (1983) onde eles propuseram um algoritmo que explora o espaço de busca de um problema combinatório, simulando o processo de recozimento de materiais.

Existem poucas abordagens multiobjetivo para o SA. A seguir veremos duas abordagens multiobjetivo para o SA.

#### 3.4.2.1. Multi-Objective Simulated Annealing (MOSA)

Este algoritmo foi proposto por Ulungu et al (1995) usando o padrão original do *Simulated Annealing*, com modificação no critério de aceitação do vizinho. Para aceitar um vizinho, o MOSA usa o conceito de Pareto.

O algoritmo apresenta uma boa eficiência, visto que pode encontrar um pequeno grupo de Pareto em um espaço curto de tempo. O MOSA pode ser descrito, de forma geral, em apenas três passos, como podemos ver no pseudocódigo da Figura 15.

```
Algoritmo MOSA
S = So
T = To
Repita
  Gera um vizinho S' = V(S)
  Se C(S') domina C(S) Então
    Aceita S'
  Senão Se C(S) domina C(S') Então
    S' somente será aceita dependendo da probabilidade
    Pt(C(S),C(S'),T)
  Senão Se C(S') não-domina C(S) e C(S) não-domina C(S') Então
    Aceita S'
  T = annealing(T)
Até uma temperatura seja satisfeita
```

Figura 15: Pseudocódigo do Multi-Objective Simulated Annealing.

A função de probabilidade $P_t(C(S),C(S'),T) = min\{exp(-c(i,j)/T),0\}$, onde o termo *c(i,j)*, da função, representa o critério de custo avaliado e *T* é a temperatura corrente do algoritmo.

O critério de custo é a forma, que é usada para comparar as duas soluções. Esse critério pode ser feito tanto analisando o custo em uma função específica, quanto à média das diferenças etc. Para obter mais detalhes sobre o critério de custo, ver Nam e Park (2000).

#### 3.4.2.2. Pareto Simulated Annealing (PSA)

Esse algoritmo, proposto por Czyżak e Jaszkiewicz (1997), possui um vetor de solução, que guarda todas as soluções não-dominadas encontradas. A cada nova solução não-dominada, que é encontrada, esse vetor é varrido para tentar encontrar soluções que sejam dominadas por essa nova solução. Caso isso aconteça, a solução dominada é retirada da lista e a nova é adicionada em seguida.

O critério de parada do algoritmo é um número determinado de soluções não-dominadas, ou quando não se consegue mais achá-las.

### 3.4.3. Busca Tabu

Essa metaheurística foi publicada por Glover (1990) para problema de roteamento. A busca tabu obtém ótimos resultados comparados com os algoritmos genéticos. Apesar de obter ótimos resultados em solução de

problemas de otimização, são poucos os trabalhos publicados que falam de busca tabu para a solução de POMO.

A seguir, iremos mostrar o MOTS, como exemplo de um algoritmo multiobjetivo, que usa a busca tabu.

#### 3.4.3.1. Multi-Objective Tabu Search (MOTS)

Esse método faz uma otimização em várias linhas de frente, utilizando uma busca tabu multiobjetivo. Nessa técnica proposta por Hansen (1997), ele forma um conjunto com as soluções correntes e outro com as soluções não-dominadas, onde, para cada solução corrente, é explorado um número definido de vizinhos através de uma função de vizinhança, onde o melhor é aceito se melhorar a solução atual; isso através de uma soma ponderada e, se não estiver na lista tabu, ou se for uma solução não-dominada. Caso a nova solução não seja inferior, ela é armazenada no conjunto de soluções não-dominadas e é feita a verificação para eliminar as possíveis soluções dominadas.

Algumas questões foram colocadas pelo autor, como sendo pontos a se estudar. Entre elas, podemos destacar:

- O que a lista tabu vai guardar? – o próprio autor sugere guardar, como exemplo, o valor de uma das funções.
- Quantas soluções correntes explorar? – ele mostra um *trade-off* entre o número de soluções, o esforço computacional e a qualidade das soluções.
- Um mecanismo de diversificação mais eficiente – para evitar que a busca se prenda a certas regiões do espaço de busca do problema.

Critério de parada – vários critérios podem ser usados, mas o autor sugere que se tenha um conjunto não-dominado base para obter uma comparação mais eficiente.

### 3.4.4. Algoritmos Evolucionários

Dos algoritmos evolucionários, os algoritmos genéticos são os mais

estudados. Isso é devido ao sucesso dos seus algoritmos na solução de problemas de otimização combinatória, onde os algoritmos retornam no final, um conjunto de soluções teoricamente evoluídas, ou seja, boas soluções que são as que realmente são procuradas.

Os algoritmos genéticos foram desenvolvidos para simular um processo evolutivo através da adaptação dos indivíduos de uma população. Esses algoritmos obtiveram um grande êxito quando aplicados para encontrarem máximos e mínimos de funções matemáticas complexas e, daí em diante, só cresceu, conseguindo excelentes performances nas mais diversas áreas da otimização.

Em relação aos problemas de otimização multiobjetivo, os algoritmos genéticos apresentam um ponto fortíssimo, pois não há a necessidade de se fazer modificações profundas nos operadores, principalmente os de cruzamento e mutação, já que estes só dependem da forma com a qual a solução é representada.

# CAPÍTULO 4

# ALGORITMOS EVOLUCIONÁRIOS MULTI-OBJETIVO

Neste capítulo, apresentamos a aplicação dos algoritmos evolucionários nos problemas de otimização multiobjetivo, descrevendo alguns dos diferentes modelos existentes. A Tabela 1 lista os principais modelos evolucionários, para otimização multiobjetivo, e seus autores (Deb, 2001 e Kunkle, 2005).

Tabela 1: Modelos Evolucionários.

| Sigla | Nome do Modelo | Autores |
|---|---|---|
| VEGA | Vector Evaluated Genetic Algorithm | Schaffer, 1984 |
| WBGA | Weight Based Genetic Algorithm | Hajela e Lin, 1992 |
| MOGA | Multiple Objective Genetic Algorithm | Fonseca e Fleming, 1993 |
| NSGA | Non-Dominated Sorting Genetic Algorithm | Srinivas e Deb, 1994 |
| NPGA | Niched-Pareto Genetic Algorithm | Horn et al., 1994 |
| TDGA | Thermodynamical Genetic Algorithm | Kita et al., 1996 |
| PPES | Predator-Prey Evolution Strategy | Laumanns et al., 1998 |
| SPEA | Strenght Pareto Evolutionary Algorithm | Zitzler e Thiele, 1998 |
| MOMGA I | Multiobjective Messy Genetic Algorithm I | Veldhuizen, 2000 |
| MOMGA II | Multiobjective Messy Genetic Algorithm II | Veldhuizen, 2000 |
| NSGA II | A Fast Elitist Non-Dominated Sorting Genetic Algorithm | Deb et al., 2000 |
| PAES | Pareto-Archived Evoltionary Strategy | Knowles e Corne, 2000a |
| M-PAES | A Memetic Pareto-Archived Evoltionary | Knowles e Corne, |

| | | |
|---|---|---|
| | Strategy | 2000b |
| PESA I | Pareto Envelope-Base Selection Algorithm I | Corne et al., 2000 |
| NPGA II | Niched-Pareto Genetic Algorithm II | Erickson et al., 2001 |
| Micro-GA | Multi-Objective Micro-Genetic Algorithm | Coello e Pulido, 2001 |
| REMOEA | Rudolph's Elitist Multi-Objective Evolutionary Algorithm | Rudolph, 1998 |
| SPEA 2 | Strenght Pareto Evolutionary Algorithm 2 | Zitzler et al., 2001 |
| PESA II | Pareto Envelope-Base Selection Algorithm II | Corne et al., 2001 |
| ISPEA | Immunity SPEA | Hongyun e Sanyang, 2003 |
| CAEP | Cultural Algorithm with Evolutionary Programming | Coello e Becerra, 2003 |
| Micro-GA 2 | Multi-Objective Micro-Genetic Algorithm 2 | Pulido e Coello, 2003 |
| MPGA | Multi-Population Genetic Algorithm | Cochran et al., 2003 |
| SPEA 2+ | Improving the Performance of the Strength Pareto Evolutionary Algorithm 2 | Kim et al., 2004 |
| MOPSO | Multi-Objective Particle Swarm Optimization | Coello et al., 2004 |
| ParEGO | Pareto Ecient Global Optimization | Knowles, 2004 |

## 4.1. Vector Evaluated Genetic Algorithm (VEGA)

O modelo VEGA foi a primeira implementação do algoritmo genético, para otimização multiobjetivo. Essa implementação foi proposta por Schaffer em 1984 (Schaffer, 1984). Tal modelo é baseado no software, de domínio público, *GENESIS*, onde Schaffer alterou o procedimento de seleção, criando um laço, fazendo com que o procedimento seja repetido para cada objetivo, separadamente.

Um dos problemas do VEGA é que esse algoritmo não obtém boa diversidade nas soluções da Fronteira de Pareto. Esse problema é devido à seleção independente dos indivíduos, pois essa seleção provoca a especialização da população, fazendo a população convergir na direção das soluções ótimas individuais após um grande número de gerações.

## 4.2. Multiple Objective Genetic Algorithm (MOGA)

O modelo MOGA (Fonseca e Fleming, 1993) foi o primeiro algoritmo a dar ênfase ao conceito de dominância de Pareto e à diversidade das soluções.

Esse algoritmo propõe uma classificação dos indivíduos de uma população de acordo com o número de indivíduos que o dominam. A cada individuo é associado um valor de *ranking,* que é igual ao número de indivíduos que domina mais um.

$$r_i = 1 + p(i)$$

Onde *p(i)* é a função que retorna o número de indivíduos, da população, que domina o indivíduo *i*. Assim, os indivíduos não-dominados possuem *ranking* igual a 1. Pelo menos, um indivíduo da população possui valor de $r_i = 1$, o valor máximo de $r_i$ não é maior do que o tamanho da população.

Depois disso, são feitos os seguintes procedimentos:

- Ordena-se a população de acordo com o *ranking*.
- Associado a cada indivíduo um *fitness* construído pela interpolação dos melhores valores das funções dos cromossomos de ranking igual a 1 e os de piores valores dos piores *rankings*.
- Depois, é feito um nivelamento entre os cromossomos que possuem um mesmo *ranking*, para evitar valores distorcidos para o *fitness*.

Esse algoritmo é um dos mais utilizados, devido a sua facilidade de ser implementado. Além de tudo, alguns autores comentam que o critério de nivelamento do *fitness* faz com que haja uma convergência muito rápida para uma região do espaço de soluções.

A Figura 16a mostra um conjunto de indivíduos e a Figura 16b mostra os valores de $r_1$, para os indivíduos.

Coello afirma que o MOGA tem desempenho melhor quando comparado com outros algoritmos não elitistas (Coello, 2001).

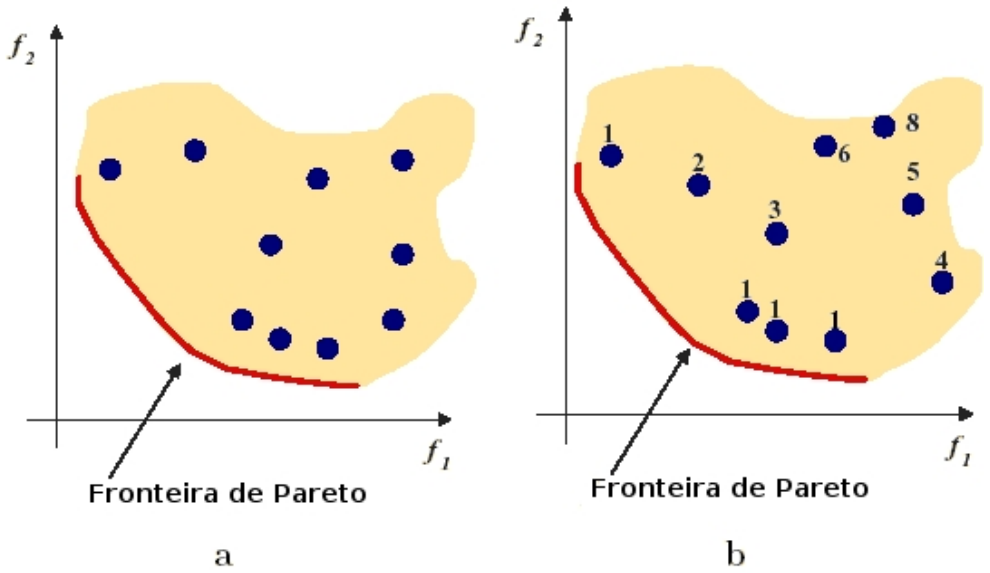



Figura 16: Cálculo do ranking do algoritmo MOGA.

O MOGA utiliza também, na sua implementação, um método de formação de nichos para distribuir a população através da região Pareto ótimo, além de compartilhar os valores de aptidão (Castro, 2001). A Figura 17 ilustra um conjunto de soluções distribuídas em vários nichos.

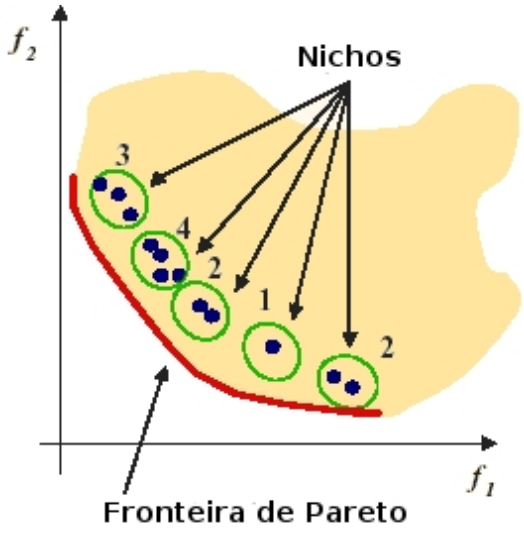



Figura 17: Conjunto de soluções agrupadas em nichos.

## 4.3. Non-Dominated Sorting Genetic Algorithm (NSGA)

O modelo NSGA (Srinivas e Deb, 1994), como o próprio nome diz, faz uma ordenação dos indivíduos da população de acordo com o critério da não-dominância. A idéia do algoritmo é utilizar um procedimento de seleção por ordenamento para enfatizar as soluções não-dominadas correntes, juntamente com um método voltado para a criação de nichos para manter a diversidade da população.

Antes de ser executado o procedimento de seleção, os indivíduos são separados em categorias, com base num nível de não-dominância dos indivíduos, isto é, todos os indivíduos não-dominados da população corrente irão receber um valor alto de aptidão. A cada passo desse procedimento são armazenados, na categoria corrente, os indivíduos que não são dominados por qualquer outro da população. Logo após, retiram-se dessa classificação os indivíduos já classificados e executa-se novamente o procedimento, com um valor de aptidão um pouco menor que o pior valor de aptidão anterior. Esse procedimento é executado até que todos os indivíduos da população tenham seus grupos.

O critério de escolha na seleção se baseia nessa classificação. Se os dois pertencem ao mesmo grupo, então, se pode escolher qualquer um dos dois mediante qualquer um dos métodos. O grande problema desse algoritmo é o alto custo computacional da função de classificação.

## 4.4. Niched Pareto Genetic Algorithm (NPGA)

O NPGA (Horn et al., 1994) compara somente uma parte dos indivíduos, de uma população, ao invés de comparar todos os indivíduos. Segundo seus idealizadores, métodos de seleção que utilizam toda a população são lentos e também muito difíceis, já que não é fácil encontrar uma população que domine várias outras.

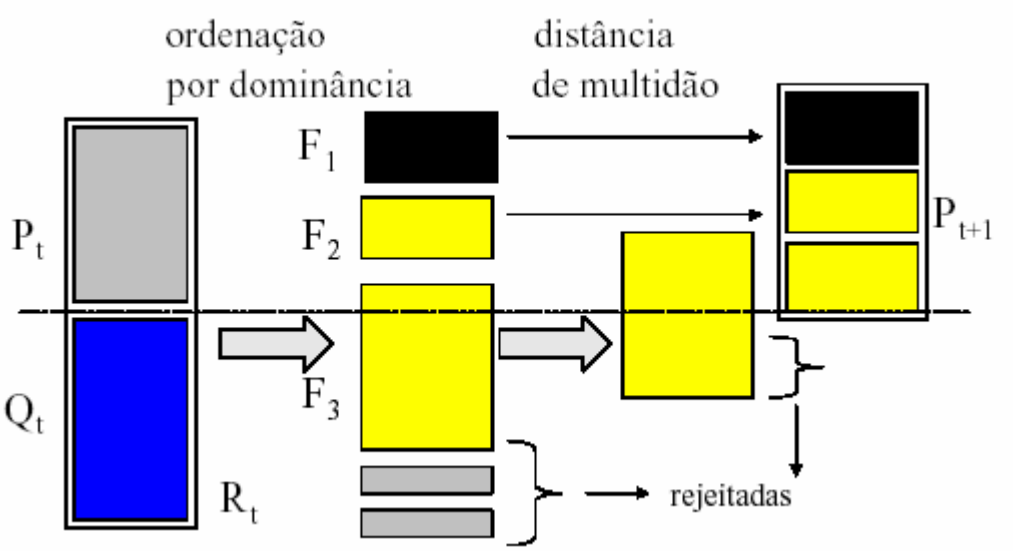



Figura 18: Esquema do modelo NSGA-II.

A proposta do NPGA é fazer a seleção através de um torneio, onde o procedimento de seleção vai agir em uma porção restrita da população (nicho). O nicho é composto por um número especifico de indivíduos ($t_{dom}$), que são tomados aleatoriamente. Em seguida, dois indivíduos são retirados da população para a seleção de um vencedor conforme o seguinte procedimento: comparam-se dois indivíduos com todos os membros do nicho, para determinação da dominância segundo as funções objetivo. Se um deles é não-dominado e o outro é dominado, o individuo não-dominado é selecionado, e declarado o vencedor, mas, caso contrário, o algoritmo utilizará o critério de *fitness* sharing (Goldberg e Richardson, 1987), no qual se calcula uma distância de cada valor da função objetivo. Isso com os melhores indivíduos do torneio.

O sucesso do NPGA é altamente dependente do parâmetro $t_{dom}$. Se um tamanho apropriado para esse parâmetro for escolhido, o algoritmo poderá encontrar pontos não-dominados (Ótimo de Pareto). Caso $t_{dom}$ seja pequeno, poderá levar o algoritmo a uma convergência prematura (Horn et al., 1994).

## 4.5. Strength Pareto Evolutionary Algorithm (SPEA)

O SPEA (Zitzler e Thiele, 1998) é um algoritmo evolucionário multiobjetivo elitista com conceitos de não-dominância.

O funcionamento desse algoritmo está associado à manutenção de uma população externa, ou repositório de informações, que, a cada geração, armazena um conjunto de soluções não-dominadas. Para cada individuo do repositório, também é associado um fator força (aptidão), que receberá o número de indivíduos da população, que são dominados por ele, Esse fator representará o *fitness* do indivíduo.

O próximo passo, do algoritmo, é a construção de uma população temporária. A construção dessa população é feita a partir de um torneio entre os indivíduos da população e do repositório. Depois de construída, os operadores genéticos são aplicados para gerar a nova população. Se a nova população for não-dominada, ela será inserida no repositório, mas, caso o repositório esteja cheio, o algoritmo exclui alguns indivíduos.

## 4.6. NSGA II

O modelo NSGA-II (Deb et al., 2000) resolve os problemas existentes no modelo NSGA, constituindo uma versão baseada em um ordenamento elitista por não-dominância.

O NSGA-II trabalha como os algoritmos genéticos convencionais, com uma população pai $P$ para gerar uma população filha $Q$. Tanto $P$ como $Q$ têm um tamanho igual a $N$. A população $P_0$, que é ordenada por não-dominância, é gerada na primeira iteração do algoritmo. Depois que a população é gerada, são aplicados os operadores de seleção por torneio, cruzamento e mutação para poder obter a população filha $Q_0$.

O algoritmo NSGA-II trabalha com a população $R_n$, $n= 1, 2, ..., N$, onde $R_n = P_n \cup Q_n$, com $|R| = 2N$. Depois que é gerada, a população $R_n$, é aplicado o procedimento de ordenamento por não-dominância sobre ela, obtendo-se as fronteiras $F_1$, $F_2$, ..., $F_j$, que são ordenadas em ordem decrescente em relação ás suas distâncias. Essas distâncias são calculadas por um método chamado de distância de multidão (*crowding distance*). Depois de obtidas as distâncias de todos os conjuntos, eles são inseridos na nova população $P_{n+1}$.

O preenchimento das novas populações $P_{n+1}$ é iniciado pelas soluções $F_1$, depois $F_2$ e assim por diante. Todos os conjuntos $F_j$ devem ser inseridos na sua totalidade em $P_{n+1}$, o que deve acontecer enquanto $P_{n+1} + |F_j| \leq N$. Caso tente

inserir um conjunto $F_j$ que seja $|F_j| > N - P_{n+1}$, então o algoritmo NSGA-II escolhe as soluções de $F_j$ que estão mais espalhadas. Para entender melhor uma iteração do NSGA-II, veja a Figura 18 (Deb, 2001).

## 4.7. Pareto Archived Evolution Strategy (PAES)

O PAES (Knowles e Corne, 2000a) possivelmente seja o algoritmo mais simples capaz de gerar o conjunto de soluções Pareto ótimo (Kunkle, 2005). Ele possui três formas, (1 + 1) – PAES, (1 + λ) – PAES e (μ + λ) – PAES. A notação refere-se a: tamanho da população e número de novas soluções por geração.

As formas (1 + 1) – PAES e (1 + λ) – PAES executam somente busca local, ou seja, só pode manter uma solução corrente, em vez de uma população de pesquisa. Já a forma (μ + λ) – PAES mantém uma população.

O PAES introduz um novo procedimento *crowding*. Ele é superior ao anterior método de *niching* de duas formas.

1. O seu custo computacional é menor;
2. É adaptativa e não exige a crítica fixação de um nicho de tamanho parâmetro.

O desempenho comparável deste simples algoritmo para outros mais complexos é atribuído ao uso de um arquivo de soluções não dominadas, o que era raro na época, mas foi integrada na maioria dos estados da arte dos algoritmos.

## 4.8. A Memetic Pareto Archived Evolution Strategy (M-PAES)

O modelo M-PAES foi introduzido por Knowles e Corne (2000b), e combina a estratégia de busca local usada no PAES e estratégia de recombinação.

As fases de busca local e global do M-PAES são parcialmente independente, e cada uma mantém os seus próprios arquivos de soluções não

dominadas. O M-PAES é altamente elitista tanto no método de busca local quanto na recombinação.

O M-PAES é um algoritmo genético híbrido, e o grande sucesso dos algoritmos híbrido depende da convexidade global do espaço de busca.

## 4.9. Pareto Envelope-based Selection Algorithm (PESA I)

O modelo PESA I (Corne et al., 2000) incorpora idéias do SPEA e do PAES. A principal idéia do PESA I é a integração da seleção e do gerenciamento da diversidade em uma técnica.

Os resultados de desempenho do PESA I foram comparados ao SPEA e PAES. Seis diferentes funções foram executadas e o melhor algoritmo para cada função foi identificado (Kunkle, 2005):

- O PESA I foi o melhor em 3 funções e empatou com o SPEA, em 2 funções;
- O SPEA foi o melhor em 1 função e empatou com o PESA I, em 2 funções;
- PAES foi claramente o pior dos algoritmos testados.

## 4.10. NPGA II

O NPGA II foi introduzido por Erickson et al. (2001). A principal melhoria do NPGA II sobre o NPGA é o fato de utilizar o grau de dominação, para determinar a pontuação no torneio de seleção. Este método de decisão é determinístico, ou contrario do NPGA que é um método probabilístico. Veja em Fonseca e Fleming (1993) mais informações sobre o grau de dominação.

## 4.11. Micro-Genetic Algorithm (Micro-GA)

O modelo Micro-GA (Coello et. al., 2001) usa populações muito pequenas (4 indivíduos) e uma reinicialização de processo. Com isso esse algoritmo propõem ser um algoritmo muito rápido e com baixo custo computacional.

O Micro-GA pode ser o primeiro o micro-GA para otimização multiobjetivo, embora a forma baseada em população do PAES poderia considera o PAES como um micro-GA (Kunkle, 2005).

Os autores listam três tipos de elitismo:

1. Armazenamento de soluções não dominadas em um arquivo externo;
2. *Carrying* sobre os melhores indivíduos entre as reinterações da população;
3. Permitir que os indivíduos das soluções não dominadas participe da busca ativa.

## 4.12. SPEA 2

O modelo SPEA 2 (Zitzler et al., 2001) é um algoritmo que utiliza o elitismo através de uma população externa *E* onde são guardadas as soluções não-dominadas. O que diferencia esse algoritmo do SPEA é o calculo do *fitness* e o parâmetro de entrada *N*, que fixa o tamanho da população externa.

No início do algoritmo, são criadas as populações $P_0$, que é uma população aleatória, e *E*, que é uma população externa. Depois, é obtido o valor de aptidão para as soluções de $Q = P \cup E$, Esse valor é obtido em várias etapas. Na primeira etapa, um valor de aptidão $s_i$ (*strenght fitness*) é encontrado, o valor $s_i$, sendo o número de soluções que *i* domina em Q. E, na segunda etapa, calcula-se o valor $r_i$ (*raw fitness*), que significa a soma dos $s_j$ das soluções *j* que são dominadas por *i* em *Q*. Assim, podemos dizer que soluções não-dominadas têm $r_i = 0$ e soluções que são dominadas por muitas soluções em Q tem $r_i$ muito alto. A Figura 19 apresenta um conjunto de soluções e seus valores ($s_i$, $r_i$).

Esse mecanismo, mesmo permitindo uma ordenação por dominância, pode falhar caso existam muitas soluções não-dominadas, onde existiram muitas soluções $r_i = 0$, sendo, assim, difícil enfatizar a preferência de uma solução sobre uma outra. Mas, para solucionar esse problema, os autores do SPEA 2 usaram uma informação de densidade para cada solução *i* de *Q*. Essa

densidade $d_i$ é inversamente proporcional à distância a seu *k-vizinho* mais próximo $\sigma^k_i$, onde $k = \sqrt{|Q|}$. O valor de $d_i$ está dentro do intervalo aberto *(0,1)*.

Com os valores de $r_i$ e $d_i$, é possível calcular o valor de $F_i$, que é a aptidão da solução. Para as soluções não-dominadas $F_i < 1$, mas, para as demais soluções, $F_i \geq 1$.

O próximo passo do algoritmo, depois de calcular as aptidões, é copiar as soluções não-dominadas de *Q* para a nova população externa $E_{n+1}$, e, depois, verificar as 3 possíveis situações:

1. $|E_{n+1}| = N$ , não faz nenhuma modificação sobre $E_{n+1}$.
2. $|E_{n+1}| < N$, ordena *Q* por $F_i$ e copia as primeiras $N - |E_{n+1}|$ soluções *i* de *Q* tal que $F_i \geq 1$.
3. $|E_{n+1}| > N$, nesse caso é usado um algoritmo de corte[1] sobre $E_{n+1}$.

Feitos todos esses passos, finalmente, o algoritmo irá realizar o processo de seleção por torneio, cruzamento e mutação, sobre $E_{n+1}$, para poder gerar a nova população $P_{n+1}$.

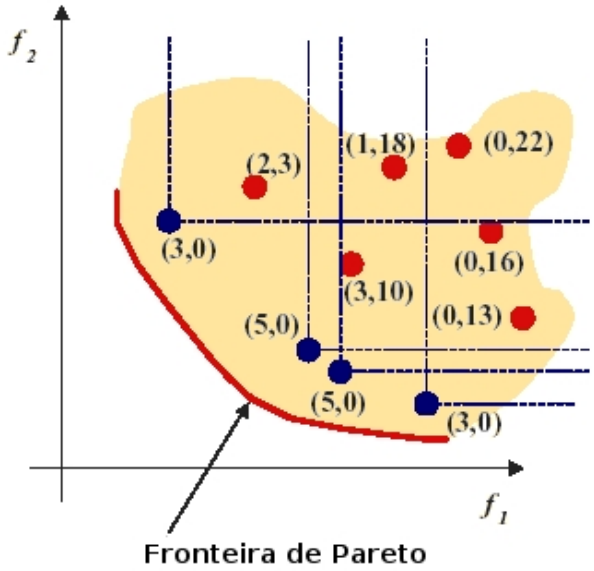


Figura 19: Esquema para o cálculo de aptidão no algoritmo SPEA 2.

[1] O algoritmo de corte, do SPEA 2, reduz o tamanho de $E_{n+1}$ para *N*.

## 4.13. PESA II

A principal melhoria do PESA II (Corne et al., 2001) comparado ao PESA I é a utilização da região baseada em seleção. Em vez de selecionar os indivíduos, *hyperboxes* no espaço de objetivos são selecionados. O *hiperboxes* representam um conjunto de possíveis soluções.

Veja a referencia original para obter bons argumentos de que a região baseada em seleção é melhor que a região baseada em indivíduos.

Os resultados de desempenho do PESA II foram comparados ao PAES, SPEA e PESA I. E ficou demonstrado que é superior a todos eles.

## 4.14. Cultural Algorithm with Evolutionary Programming (CAEP)

O modelo CAEP proposto por Coello et. al. (2003) é o primeiro algoritmo da espécie algoritmo cultural multiobjetivo, adotando Pareto ranking e elitismo.

Algoritmo cultural trabalha em dois espaços, o espaço normal de população, e mais um espaço de convicção. Conhecimentos adquiridos pela evolução são armazenados no espaço de convicção e é utilizado para influenciar a evolução.

O espaço de convicção nesse caso contém duas partes:

1. A parte fenotípica normativa: o limite inferior e superior para cada função objetivo;
2. Uma grade para evitar soluções aglomeradas (uma variação na grade proposta por Knowles e Corne (2000a)).

## 4.15. Micro-GA 2

O Micro-GA 2 (Pulido e Coello, 2003) possui duas principais estágios (etapas): prospecção e exploração. A principal diferencia entre as duas etapas está na ponderação relativa de mutação e cruzamento. Na etapa de prospecção, a mutação domina; na etapa de exploração, o cruzamento domina.

O Micro-GA 2 foi comparado com Micro-GA, NSGA II e PAES, e os resultados de desempenho foram (Kinkle, 2005):

- O Micro-GA2 é consideravelmente superior ao Micro-GA e PAES;
- O NSGA II supera o Micro-GA em algumas funções, ou seja, o NSGA II pode ser ainda superior ao Micro-GA 2.

## 4.16. Multi-Population Genetic Algorithm (MPGA)

O modelo MPGA foi introduzido por Cochran et al. (2003). Esse modelo usa um processo de duas etapas, onde cada etapa é baseada em um algoritmo multiobjetivo:

1. baseada no MOGA.
2. baseada no VEJA.

Não foi encontrada nenhuma comparação do MPGA com outros algoritmos evolucionários multiobjetivo. Por ser baseado em dois algoritmos inferiores aos melhores algoritmos multiobjetivo atuais, então não é provável que o MPGA obtenha soluções melhores (Kinkle, 2005).

## 4.17. SPEA 2+

O SPEA 2+ (Kim et al., 2004) é a melhora do SPEA, acrescentando duas coisas:

- Um mecanismo mais eficiente de cruzamento;
- Um algoritmo para manter a diversidade, tanto no espaço de objetivos quanto no espaço de variáveis.

Especificamente essas modificações são conseguidas com as seguintes operações:

- Cruzamentos entre vizinhos: cruza indivíduos próximos uns dos outros no espaço objetivo;
- Elitismo forte: todos indivíduos (soluções não dominadas) participar da seleção;

- Manter dois arquivos: um para manter a diversidade no espaço objetivo, o outro no espaço de decisão.

O SPEA 2+ apresenta uma maior diversidade no espaço de decisão comparado ao SPEA 2.

## 4.18. Pareto Efficient Global Optimization (ParEGO)

O ParEGo (Knowles, 2004) usa uma abordagem híbrida: usa tanto uma busca evolutiva e um modelo interno detalhado. O modelo é usado para selecionar áreas do espaço de busca que pode fornecer melhores soluções e melhorar o modelo interno.

Converte busca multiobjetivo em monoobjetivo, usando diferentes pesos nos objetivos.

## 4.19. Particle Swarm Optimization

Particle Swarm Optimize (PSO) são algoritmos de otimização baseados no comportamento social dos bandos de pássaros (Kennedy and Eberhart, 1995). O PSO é um processo baseado na hipótese de que os indivíduos de uma população (*swarm*) podem lucrar com as experiências passadas e as experiências de outros indivíduos (*partícula*). Em outras palavras, o PSO é um processo de busca baseado em uma população onde indivíduos são partículas agrupadas em *swarm*. Cada partícula no *swarm* representa uma solução para o problema de otimização.

Em um sistema de PSO, cada partícula explora o espaço de busca para ajustar sua posição de acordo com sua experiência e a de sua vizinhança. Durante essa exploração, cada partícula tem acesso a duas informações importantes que são: a melhor solução potencial encontrada por ela e a melhor solução potencial encontrada pelos seus vizinhos. Essas informações são usadas para dirigir a busca a uma solução ótima.

As partículas de *swarm* utilizam a topologia anel (Kennedy, 1997), como podemos ver na Figura 20.

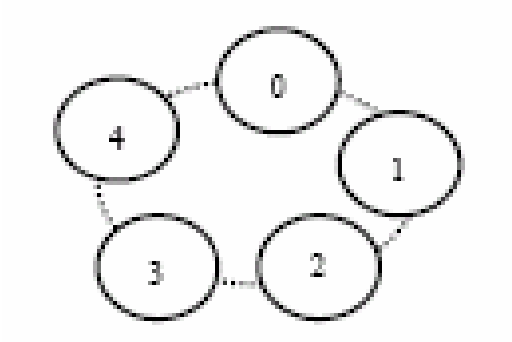



Figura 20: Topologia de um *swarm.*

A Figura 20 ilustra a topologia de um *swarm* de cinco partículas, onde a vizinhança da partícula *i* consiste nas partículas *i-1*, *i* e *i+1*. Por exemplo, a vizinhança da partícula 4, da Figura 19, consistiria nas partículas 3, 4 e 0.

Uma partícula é composta por três elementos (vetores) *x*, *p* e *v* (Kennedy, 1997). O vetor $x = [x_{i1}, x_{i2}, ..., x_{im}]$ contém as soluções potenciais correntes da partícula. O vetor $p = [p_{i1}, p_{i2}, ..., p_{im}]$ contém as melhores soluções potenciais descobertas por uma partícula. Já o vetor $v = [v_{i1}, v_{i2}, ..., v_{im}]$ é conhecido como vetor velocidade, pois é usado para determinar a próxima solução potencial a ser avaliada. Além dos vetores *x*, *p* e *v*, a partícula tem dois valores de aptidão (*fitness*). O Valor $\chi_i$ representa o *fitness* atribuído a $x_i$ pela função objetivo e o valor $\rho_i$, que é o *fitness* atribuído ao vetor *p*. O comprimento dos vetores vai depender do número de parâmetros do problema (Moore and Chapman, 1999).

O PSO inicia, aleatoriamente, os vetores *x* e *v*. Inicialmente, o vetor *p* é igual ao vetor *x*. Toda vez que o vetor *x* atualizar suas partículas, $\chi_i$ é comparado com $\rho_i$ e atualiza *p*. Por exemplo, se o problema for de minimização, $p_i$ será igual a $x_i$ se $\chi_i$ for menor que $\rho_i$. Já, para um problema de maximização, $p_i$ será igual a $x_i$ se $\chi_i$ for maior que $\rho_i$. Com isso, o vetor *p* contém sempre as melhores soluções potenciais descobertas por uma partícula.

### 4.19.1. Multi-Objective Particle Swarm Otimizer

Para adaptar o PSO para solucionar POMO, o vetor *p* foi modificado, para ser uma lista de soluções, para manter todas as soluções não-dominadas

encontradas por uma partícula, na exploração do espaço de busca (Moore e Chapman, 1999).

O PSO multiobjetivo inicia aleatoriamente os vetores *x* e *v*. Toda vez que o vetor *x* atualizar uma das suas partículas, ele a comparará com as soluções da lista *p*, para determinar se a solução é não-dominada. Se a solução for não-dominada, ela será adicionada à lista *p*. Para assegurar que sempre conterá soluções não-dominadas, a lista *p* será atualizada constantemente (Moore e Chapman, 1999).

A exploração do espaço de busca é guiada por duas informações importantes: a melhor solução potencial descoberta por uma partícula, e a melhor solução potencial descoberta pela sua vizinhança. Com a adaptação do PSO para multiobjetivo, o vetor *p*, agora chamado de lista *p*, poderá conter soluções numerosas, as melhores soluções descobertas por um indivíduo. Para determinar a melhor solução potencial na vizinhança, Moore e Chapman (1999) compararam as soluções encontradas na lista *p* com a vizinhança, para encontrar uma, que não é dominada. O restante do algoritmo PSO multiobjetivo é idêntico ao PSO clássico.

### 4.19.2. Multi-Objective Particle Swarm Optimization (MOPSO)

O MOPSO (Coello et al., 2004) utiliza uma forma de seleção baseada em região.

Comparações de desempenho do MOPSO com NSGA II, Micro-GA e PAES mostraram que o MOPSO foi o único algoritmo no estudo capaz de cobrir toda a fronteira de Pareto das funções utilizadas (Kunkle, 2005).

# CAPÍTULO 5

# PROBLEMA DO CAIXEIRO VIAJANTE

O Problema do Caixeiro Viajante (PCV) é um dos mais tradicionais e conhecidos problemas de Otimização Combinatória, estando associado à determinação dos caminhos hamiltonianos em um grafo qualquer. O PCV pode ser definido da seguinte forma: dado um número finito de cidades, com custo de viagem entre elas, encontrar o caminho mais barato para visitar uma única vez todas as cidades e retornar à cidade inicial.

Embora tenha uma forma simples, ele é intratável e pertencente à classe NP-Árduo. Daí, a dificuldade de se desenvolver algoritmos eficientes para o problema (Campello et al, 1994).

O objetivo do Problema do Caixeiro Viajante é encontrar, em um grafo *G = (N, A)*, um caminho hamiltoniano[2] ou *tour* de peso mínimo. Para solucionar problemas dessa natureza, existem muitos métodos heurísticos, que produzem boas soluções para o problema. Isso porque não é conhecido algoritmo eficiente, que resolva o PCV.

## 5.1. Histórico

O problema matemático relacionado para o PCV foi tratado em 1857, pelo matemático irlandês Sir Willian Rowan Hamilton.

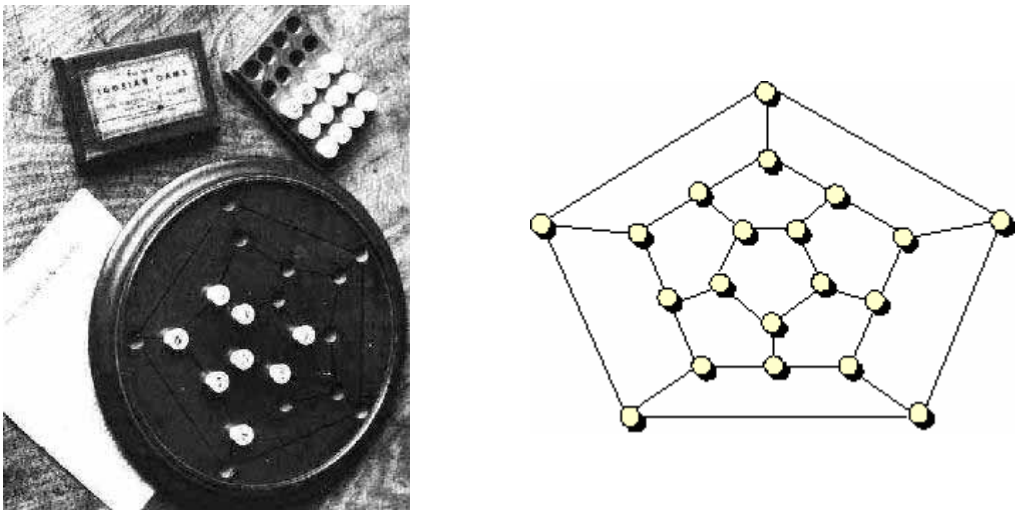
Hamilton propôs um jogo, que denominou *Around the World*. Esse jogo é montado sobre um dodecaedro, onde cada vértice estava associado a uma cidade importante da época. O desafio proposto pelo jogo consistia em

[2] Caminho hamiltoniano é um ciclo que passa por todos os vértices.

encontrar uma rota através dos vértices do dodecaedro que iniciasse e terminasse em uma mesma cidade sem nunca repetir uma visita.

Hamilton não foi o primeiro a tratar do problema do caixeiro viajante. No mesmo século, o matemático britânico Thomas Penyngton Kirkman, também tratou desse problema. Mas foi justamente o jogo de Halminton, que o divulgou. Uma discussão agradável sobre os trabalhos de Hamilton e Thomas pode ser vista em Biggs et. al. (1976).

A Figura 21a mostra o jogo de Hamilton (http://www.tsp.gatech.edu), e a Figura 20b mostra o grafo do problema (Goldbarg e Luna, 2000).



*a* *B*

Figura 21: Jogo de Hamilton.

Uma solução, para jogo de Hamilton, passou a ser denominada de ciclo hamiltoniano, em sua homenagem. A Figura 22 (Goldbarg e Luna, 2000) apresenta uma solução para o jogo.

Mas foi, na década de 1920, que o problema se tornou público pelo matemático e economista Karl Menger, juntamente com os seus colegas em Viena. Na década de 1930, ele foi utilizado pelos matemáticos da universidade de Princeton e, na década de 1940, foi usado numa aplicação voltada para a agricultura, por estatísticos. Mas, a partir da década de 1940, o PCV foi definido como um problema difícil, da otimização combinatória, tendo em vista a inviabilidade de examinar todas as soluções, uma a uma, isso devido ao seu

enorme número, outra coisa que contribuiu, na época, nessa definição, foi a falta de idéia de como resolvê-lo.

Em 1954, foi publicado por Dantzig, Fulkerson e Johnson um método para solucionar o PCV. Para mostrar o poder do seu método, os autores resolveram uma instância de 49 cidades dos Estados Unidos. Essa instância, na época, era considerada gigantesca. O artigo de Dantzig, Fulkerson e Johnson é um dos mais importantes nessa área de pesquisa, constituindo um marco nas relações entre a programação linear e a otimização combinatória. No próximo tópico, veremos as resoluções históricas para PCV.

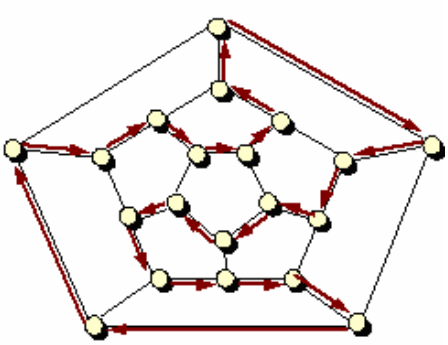


Figura 22: Uma solução para o jogo de Hamilton.

## 5.1. Resoluções Históricas

Os códigos de computador para o PCV se tornaram cada vez mais sofisticados ao passar dos anos, sendo um sinal dessas melhorias o tamanho crescente dos exemplos não trivial, que já foram resolvidos. A Tabela 2 mostra essa evolução no intervalo de 52 anos (1954-2006).

### 5.1.1. 49 Cidades

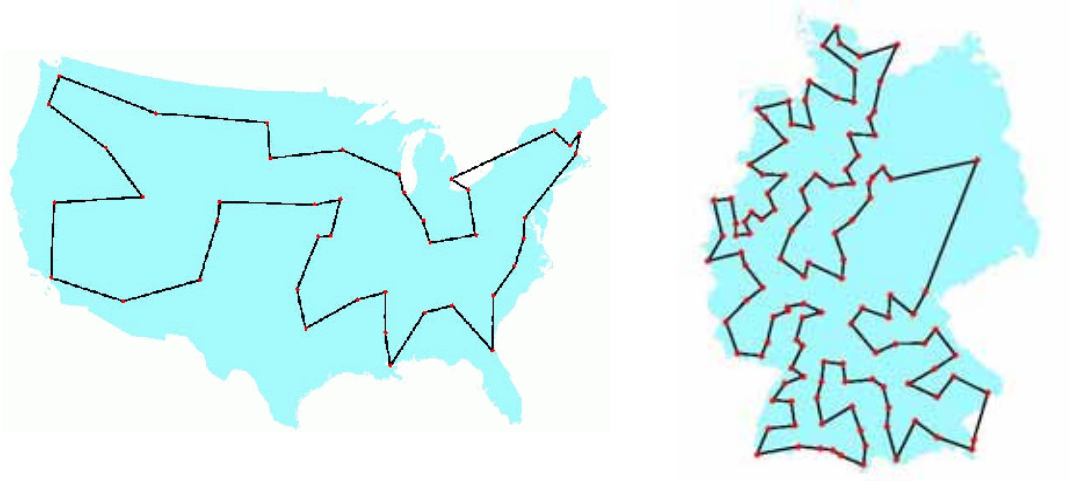
A instância de 49 cidades foi proposta em 1954 por George Dantzig, Ray Fulkerson e Selmer Johnson, para ilustrar e mostrar a força do método descrito por eles, para resolver o PCV.

Eles criaram essa instância, escolhendo uma cidade de cada um dos estados dos Estados Unidos[3] e adicionando Washington DC; o custo da viagem entre essas cidades é a distancia das estradas. A Figura 23a ilustra esse problema (http://www.tsp.gatech.edu).

[3] Os estados do Alaska e Hawaii somente tornaram-se estados em 1959.

Tabela 2: Recordes mundiais para o PCV

| Ano | Pesquisadores | Tamanho da Instância |
|---|---|---|
| 1954 | G. Dantzig, R. Fulkerson e S. Jhnson | 49 cidades |
| 1971 | M. Held e R. M. Karp | 64 cidades |
| 1975 | P. M. Camerini, L. Fratta e F. Maffioli | 100 cidades |
| 1977 | M. Grotschel | 120 cidades |
| 1980 | H. Crowder e M. W. Padberg | 318 cidades |
| 1987 | M. Padberg e G. Rinaldi | 532 cidades |
| 1987 | M. Grotschel e O Holland | 666 cidades |
| 1987 | M. Padberg e G. Rinaldi | 2392 cidades |
| 1994 | D. Applegate, R. Bixby, V. Chvátal e W. Cook | 7397 cidades |
| 1998 | D. Applegate, R. Bixby, V. Chvátal e W. Cook | 13509 cidades |
| 2001 | D. Applegate, R. Bixby, V. Chvátal e W. Cook | 15112 cidades |
| 2004 | D. Applegate, R. Bixby, V. Chvátal, W. Cook, and K. Helsgaun | 24978 cidades |
| 2006 | W. Cook, Daniel G. Espinoza e Marcos Goycoolea | 85900 cidades |



*a* *b*

Figura 23: *Tour* de 49 e 120 cidades.

### 5.1.2. 120 Cidades

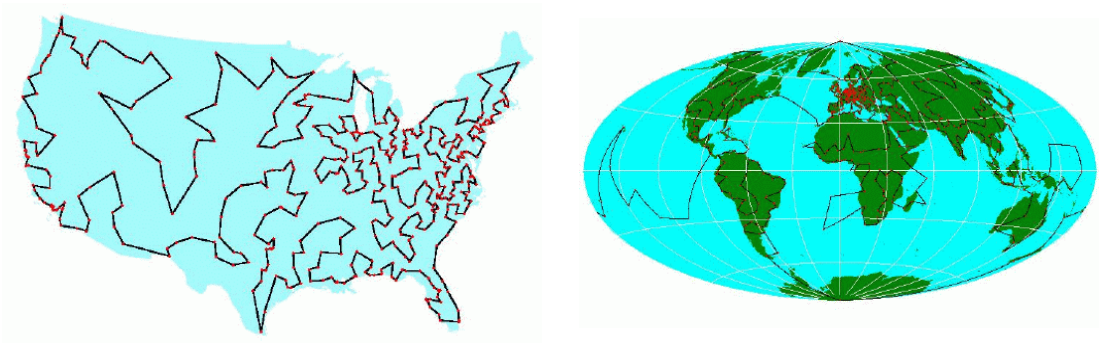
Groetschel, em 1977 encontrou um *tour* ótimo de 120 cidades, que estão no oeste da Alemanha. Veja o *tour* na Figura 23b (http://www.tsp.gatech.edu).

### 5.1.3. 532 Cidades

Padberg e Rinaldi, em 1987, encontraram um *tour* dos 532 switch AT&T localizados nos Estados Unidos. A Figura 24a ilustra esse *tour* (http://www.tsp.gatech.edu).

### 5.1.4. 666 Cidades

Groetschel e Holland, em 1987, encontraram um *tour* ótimo de 666 lugares interessantes do mundo. A Figura 24a ilustra esse *tour* (http://www.tsp.gatech.edu).



*a* *b*

Figura 24: *Tour* de 532 e 666 cidades.

### 5.1.5. 13509 Cidades

Em 1998, D. Applegate, R. Bixby, V. Chvátal e W. Cook resolveram um caso para o PCV de 13509 cidades. O caso consistia de todas as cidades localizadas nos Estados Unidos com população de no mínimo 500 habitantes, tendo sido usada uma lista do banco de dados da CIA. A Figura 25 ilustra esse *tour*, que até 2001, foi o recorde mundial para o caixeiro viajante (http://www.tsp.gatech.edu).

### 5.1.6. 15112 Cidades

Em 2001 Applegate, Bixby, Chvátal e Cook bateram o recorde mundial para o PCV, que, desde 1998, era do PCV de 13509 cidades, resolvendo um PCV para 15112 cidades da Alemanha. Esse foi o maior caso resolvido até maio deste ano. Veja na Figura 26 o *tour* do PCV de 15112 cidades (http://www.tsp.gatech.edu).

### 5.1.7. 24978 Cidades

Em maio, de 2004, foi resolvida uma instância de 24978 cidades para o PCV. Essa instância consiste das 24978 cidades da Suécia. Essa instância foi, durante dois anos, a maior instância conhecida para o PCV. O *tour* das 24978 cidades pode ser visto na Figura 27 (http://www.tsp.gatech.edu).

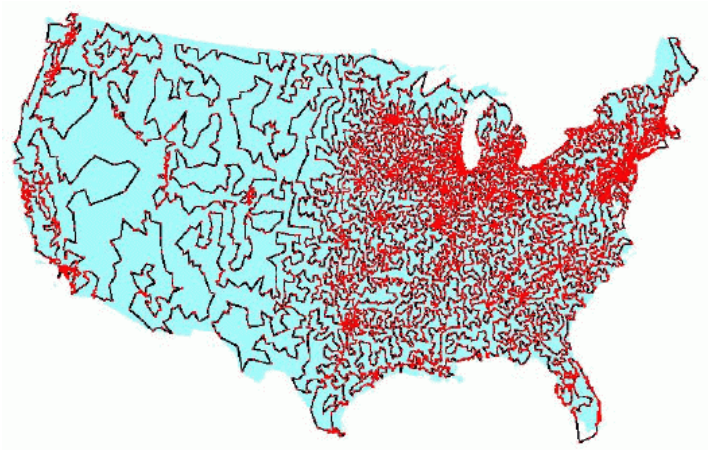


Figura 25: *Tour* de 13509 cidades.

Figura 26: *Tour* do PCV de 15112 cidades.

### 5.1.8. 85900 Cidades

O melhor resultado exato para o PCV foi obtido em abril de 2006, resolvendo a instância de 85900 cidades. Essa instância foi solucionada por William Cook, Daniel Espinoza e Marcos Goycoolea (Cook et. al., 2007). Atualmente, é a maior instância solucionada para o PCV ultrapassando, assim, o caso de 24978 cidades da Suécia, solucionada em 2004.

## 5.2. Descrição

O problema do caixeiro viajante pode ser definido da seguinte forma: Dado um grafo *G = (V, E)*, encontrar o caminho hamiltoniano mais barato (custo ótimo) para visitar todos os vértices (cidades) do grafo e retornar ao vértice inicial, passando somente uma vez em cada vértice, ou seja, um ciclo simples de tamanho |*V*| em *G* com o menor custo possível.

A função de custo é simétrica, sendo $C(d_i, d_j) = C(d_j, d_i)$, visto que trabalhamos somente com o PCV simétrico neste trabalho. O número possível de *tours*[4], para um PCV simétrico, para um grafo com N vértices é $\frac{1}{2} * (N-1)!$. Esse cálculo é simples visto que um *tour* consiste simplesmente na permutação da ordem de visitação dos N vértices, excluindo o primeiro, que pode ser arbitrário.

[4] Um caminho que sai de uma cidade visitando todas as outras e retorna à primeira cidade.

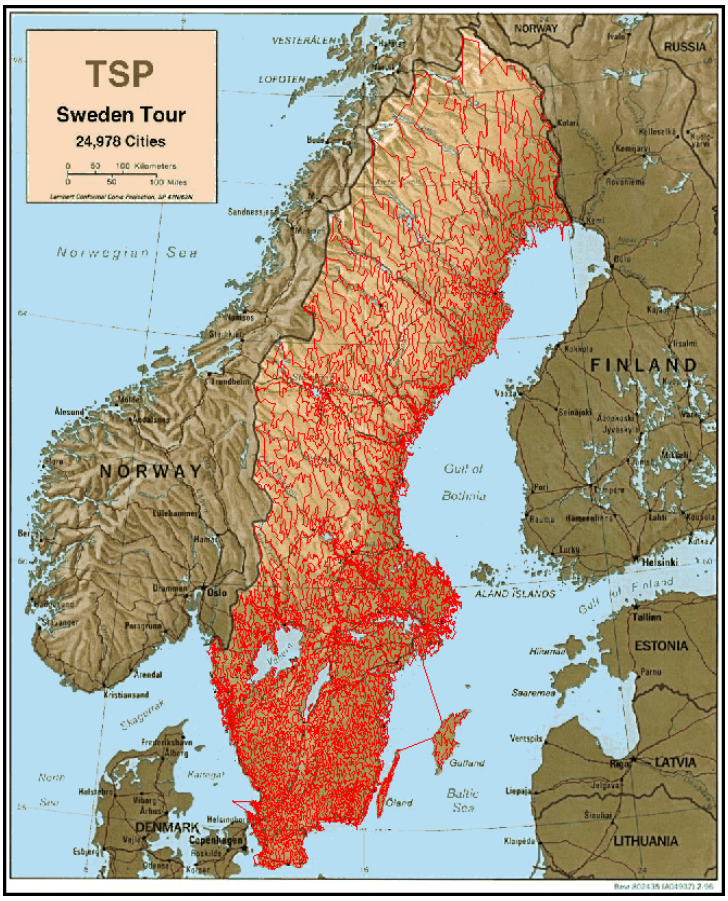
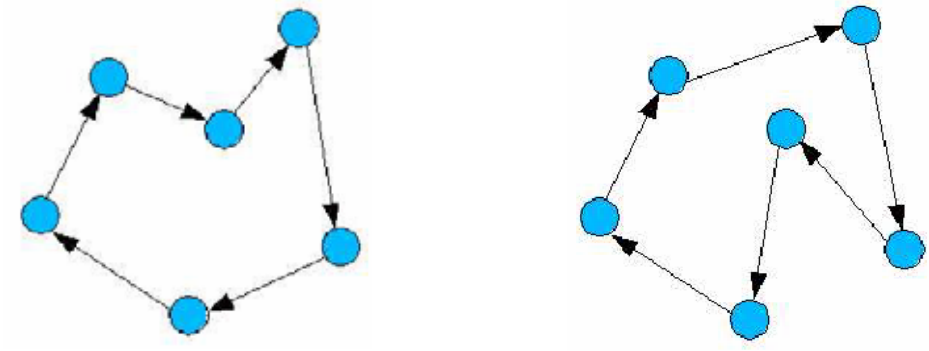



Figura 27: *Tour* do PCV de 24978 cidades.

Para entendermos melhor o PCV, podemos imaginá-lo como um planejamento turístico, para visitar várias cidades, onde a rota a ser seguida deve passar por todas as cidades apenas uma vez e, ao final, deve voltar à cidade inicial (*tour*), fechando assim o ciclo. A Figura 28 ilustra duas possíveis rotas para um PCV.



*a* *b*

Figura 28: Possíveis rotas para um PCV.

## 5.3. Formulação

Neste trabalho, iremos somente apresentar a formulação de Dantzig-Fulkerson-Johnson (DFJ), por ser tida como canônica e, também, por ser uma das formulações mais difundidas na literatura especializada, além de desenvolver um modo peculiar do PCV.

Essa formulação encara o PCV como um problema de programação *0-1*, sobre um gafo *G =(V, E)*, como podemos ver na formulação abaixo.

$$\textit{Minimizar } z = \sum_{j=1}^{n}\sum_{i=1}^{n} c_{ij}\, x_{ij}$$

*sujeito a:*

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in N \qquad (1)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in N \qquad (2)$$

$$\sum_{i,\, j \in S} x_{ij} \leq |\, S\,| - 1 \qquad \forall S \subset N \qquad (3)$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \in N \qquad (4)$$

A variável binária $x_{ij}$ somente receberá o valor *1* se a aresta $(i,j) \in E$ for escolhida para pertencer ao *tour*, e 0 caso contrário. Para essa formulação, é comum assumir-se que não existe $x_{ii}$ e que se têm $n(n-1)$ variáveis binárias.

A restrição (3) é utilizada para eliminar o circuito pré-hamiltoniano que podem ser surgidos pelas restrições (1) e (2).

## 5.4. Aplicações

A importância do PCV é devida à ocorrência de diversas aplicações práticas em áreas do conhecimento como engenharia e física (Campelo, 1994). O PCV constitui, assim, uma plataforma ideal para o estudo de métodos gerais que podem ser aplicados a uma grande quantidade de problemas de otimização combinatória.

O PCV surge, naturalmente, como um subproblema em muitas aplicações logísticas e de transporte, por exemplo, o problema para arranjar o

itinerário de um ônibus escolar, que apanha crianças em uma escola. Essa aplicação tem um significado importantíssimo para a história do PCV, por ser uma das pesquisas pioneiras na década de 1940, por Merrill Flood.

Existem muitas outras aplicações interessantes em diversas áreas do conhecimento. Uma aplicação industrial interessante encontrada na literatura é o problema da perfuração de placas de circuitos impressos, onde o PCV aparece na execução eficiente das perfurações. Essas perfurações são feitas utilizando máquinas automáticas (Campelo, 1994).

Outra aplicação menos comum foi a utilizada pelos Pesquisadores do *National Institute of Health,* que usaram soluções do PCV para construir mapas híbridos, como parte do trabalho da seqüência do genoma, onde as cidades são os mapas locais e o custo de viagem é a probabilidade medida de um mapa local imediatamente seguido do outro (Agarwala et. al., 2000). Essa aplicação foi utilizada, também, por um grupo na França para desenvolver um mapa do genoma do rato (Avner et. al., 2001).

Existe, também, o projeto da *NASA* conhecido como *Starlight* para minimizar o consumo de combustível dos satélites para fazer imagens de objetos celestiais (http://www.tsp.gatech.edu).

A utilização de algoritmos heurísticos eficientes, que decomponham o grafo em problemas menores, torna-se uma imposição nas aplicações práticas do PCV.

# CAPÍTULO 6

# PROBLEMA DO CAIXEIRO VIAJANTE MULTIOBJETIVO

O problema do caixeiro viajante multiobjetivo (PCV multiobjetivo) consiste em um grafo $G = (V, E, w)$, onde $V$ é o conjunto de vértices, $E$ o conjunto de arestas e $w$ é uma função que atribui a cada aresta ($e_{ij} \in E$) um vetor ($w^1_{ij}$,..., $w^M_{ij}$). Cada elemento $w^m_{ij}$ corresponde a um determinado peso, de métricas diferentes, por exemplo, distância e custo para uma aresta de um problema biobjetivo (Paquete et. al., 2004).

A função objetivo ($f_m$) é definida como sendo o custo de um ciclo hamiltoniano, usando os valores $w_{ij}^m$.

Para um conjunto de $n$ cidades, o PCV multiobjetivo consiste em determinar o ciclo hamiltoniano que minimize as $m$ funções objetivo. Mas, como as funções, geralmente, são conflitantes entre si, esse tipo de problema irá apresentar mais de uma solução ótima (conjunto de soluções ótimas de Pareto). Este conjunto contém todas as soluções que não são dominadas por qualquer outra solução. O problema de encontrar o conjunto Pareto ótimo é NP-hard (Paquete, 2003).

Uma formulação pode ser vista abaixo.

*F é o conjunto dos ciclos hamiltonianos*
*C é um ciclo hamiltoniano*
$w^m_{ij} \geq 0$ *é o valor atribuído pelo critério (m) a aresta* $e_{ij}$.
$m = 1, ..., M$

$i, j \in 1, ..., n$

$Minimizar\ (f_1(C), f_2(C), ..., f_M(C))$

*sujeito a:*

$f_m(C) = \sum_{e \in C} w^m(e)$

$C \in F$

O problema que tratamos é simétrico, onde $w^m_{ij} = w^m_{ji}$ para todos os pares de vértices. Na Figura 29, podemos observar um grafo que representa um PCV biobjetivo.

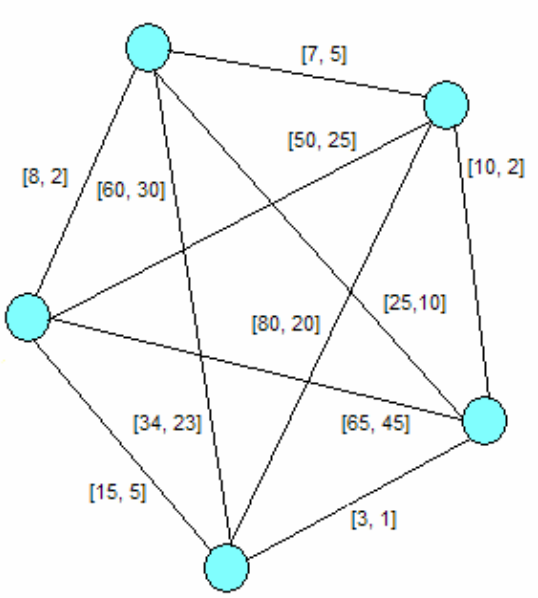



Figura 29: Grafo de um PCVMO biobjetivo.

## 6.1. Técnicas de Solução

Podemos observar, nessa sessão, algumas das técnicas usadas para solucionar o PCV multiobjetivo.

### 6.1.1. Técnica por Algoritmos monoobjetivo

É possível solucionar o problema do caixeiro viajante multiobjetivo, usando algoritmos monoobjetivo, mas, para isso, é necessário usar o artifício de se normalizar as arestas de um grafo multiobjetivo por algum tipo de norma

vetorial (Euclidiana, Tchebycheff ou outras), a fim de permitir que o problema multiobjetivo possa ser solucionado pelos algoritmos monoobjetivo.

Em Ehrgott et al (2000), ele utiliza a heurística de Christofides para solucionar o PCV multiobjetivo utilizando a técnica de normalização. Fischer e Richter (1982) e Tung (1994) usaram o método *Branch-and-Bound* para resolver o PCV multiobjetivo.

### 6.1.2. Técnica de Busca Local

Essa técnica é amplamente utilizada no problema do caixeiro viajante para encontrar soluções de boa qualidade. A maioria dos algoritmos básicos de busca local, faz busca na vizinhança de um *tour* a procura de um *tour* melhor. Se for encontrado um tour melhor o algoritmo faz a substituição. Algoritmos desse tipo dependem criticamente da definição de vizinhança.

Para essa técnica existem as seguintes heurísticas (Paquete et. al., 2004):

***2-opt***: Elimina duas arestas e substitui com um único par de arestas que não quebra o *tour*. A heurística *2-opt* faz duas trocas sempre que for possível, ou seja, só finaliza quando não pode ser aplicada nenhuma troca.

***2h-opt***: Consiste em mover duas arestas de uma única cidade de uma posição para outra. Conforme sugerido por Bentley (1992).

***3-opt***: Elimina três arestas e substitui por um outro conjunto de três arestas. A idéia aplicada na heurística *3-opt* é a mesma utilizada no *2-opt*.

As heurísticas *2-opt* e *3-op*t foi utilizada por Gupta e Warburton (1986) para solucionar o PCV multiobjetivo.

### 6.1.3. Técnicas Multiobjetivo

Alguns pesquisadores também têm usado técnicas especificas para resolver o PCV multiobjetivo. Sigal (1994) propôs uma abordagem para e Melamed e Sigal (1997) utilizaram à abordagem por $\varepsilon$-restrições para

solucionar o PCV biobjetivo. A busca tabu também foi aplicada no PCV multiobjetivo por Hansen (2000). Borges e Hansen (2002) usaram a abordagem de soma ponderada para estudar a convexidade global do PCV multiobjetivo.

### 6.1.4. Técnicas por Metaheurísticas

Pela dificuldade de solucionar um PCV multiobjetivo, não somente pela complexidade combinatória, mas também pela busca de todas as soluções eficientes, que, por sinal, crescem com o número de objetivos do problema. Muitos pesquisadores vêm utilizando metaheurísticas, como, por exemplo, algoritmos genéticos, métodos de busca local baseados em busca tabu e *simulated annealing*, para solucionar os problemas desse tipo, pois são métodos flexíveis e eficientes.

As metaheurísticas que mais se destacam nas publicações para problemas de otimização multiobjetivo, em geral, são as baseadas em algoritmos genéticos. A preferência por essas metaheurísticas se dá pelo fato de os algoritmos genéticos trabalharem com população e que podem conter informações sobre várias regiões do espaço de busca, podendo, assim, oferecer uma maior possibilidade para encontrar o conjunto de soluções Pareto ótimo.

Várias adaptações de algoritmos genéticos já foram feitas, desde o primeiro algoritmo proposto por Schaffer (1984). Mas, em geral, os algoritmos genéticos “puro” têm mostrado um desempenho muito inferior aos métodos que utilizam busca em vizinhança, como busca tabu e *simulated annealing*. Uma boa solução para resolver problemas de otimização multiobjetivo, como o PCV multiobjetivo, é utilizando a técnica de hibridação, ou seja, utilizar métodos baseados em algoritmos genéticos com outros métodos, como, por exemplo, os baseados em busca em vizinhança, pois os algoritmos híbridos são extremamente competitivos (Arroyo e Armentano, 2003).

## 6.2. Aplicação Prática

Problemas de transporte constituem exemplos de aplicações reais e práticas do PCV multiobjetivo, pois, às vezes, é necessário otimizar o custo da viagem, o tempo ou a distância percorrida.

A solução de uma aplicação desse tipo é dada pelo conjunto de soluções ótimas de Pareto. A escolha da melhor solução vai depender da preferência e da necessidade do tomador de decisão.

# CAPÍTULO 7

# ALGORITMOS EVOLICIONÁRIOS APLICADOS AO PROBLEMA DO CAIXEIRO VIAJANTE MULTIOBJETIVO

Neste capítulo, abordamos dois algoritmos evolucionários multiobjetivo, para solucionar o problema do caixeiro viajante multiobjetivo simétrico.

O tipo de caixeiro viajante multiobjetivo simétrico abordado é o biobjetivo (PCV biobjetivo), onde, para cada par de cidades (*i,j*), estão associados dois valores ($c_{ij}$ e $c'_{ij}$) para um conjunto de *n* cidades. O problema do caixeiro viajante biobjetivo consiste em determinar o ciclo hamiltoniano, que minimiza $f_1$ e $f_2$.

Como os problemas multiobjetivo possuem mais de uma solução ótima, os algoritmos evolucionários mostrados, neste capítulo, terão como resultado um conjunto de soluções para o Problema do Caixeiro Viajante (PCV) e o Problema da Mínima Latência (PML).

A seguir, serão mostrados dois algoritmos evolucionários, que podem resolver o PCV e o PML, como um problema biobjetivo: MOGA (*Multiple Objective Genetic Algorithm*) e SPEA 2 (*Strenght Pareto Evolutionary Algorithm 2*).

## 7.1 Problema da Mínima Latência

O Problema da Mínima Latência (PML) é uma variação do PCV onde o objetivo é visita os nós de um grafo de uma maneira global para minimizar os

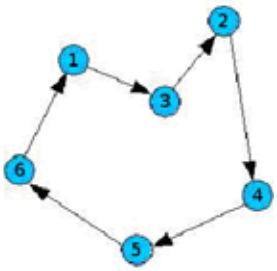
tempos de espera dos clientes localizados em cada nó do grafo. O problema foi introduzido e relacionado com o PCV em 1967 por Conway, Maxwell e Miller (Conway et. al., 1967), quando o PML era conhecido como um tipo de problema de sequenciamento. De acordo com Goemans e Kleinberg (Goemans e Kleinberg, 1998), apesar da semelhança com o PCV clássico, o PML parece ter um comportamento inferior ao PCV do ponto de vista computacional.

## 7.2 Estrutura de Representação da Solução

A estrutura adotada para representação de uma rota é um vetor de *n* posições. Associados a cada posição *i* do vetor, encontram-se os índices das cidades. A Figura 30 permite visualizar uma rota e sua representação.

A representação em vetor, além de ser uma estrutura simples, também permite operações de troca entre 3 arcos, como mostra a Figura 31.

A troca de 3 arcos é possível de ser executada em *O(1)* porque os índices das cidades envolvidas na troca podem ser acessadas diretamente no vetor, e seus valores alterados.

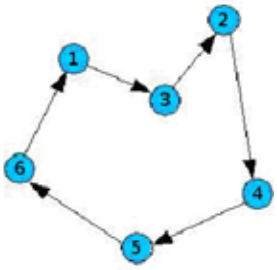



a

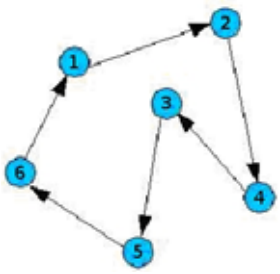
| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 3 | 2 | 4 | 5 | 6 |

b

Figura 30: Representação de uma Solução para um PCV.




a




b

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 3 | 2 | 4 | 5 | 6 |

c

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 2 | 4 | 3 | 5 | 6 |

d

Figura 31: Troca entre 3 Arcos sobre o Vetor de Solução.

## 7.3 População Inicial

A população inicial é gerada, aleatoriamente, para os dois algoritmos evolucionários aplicados neste trabalho.

## 7.4 Experimentos Computacionais

Nessa seção, são apresentados e discutidos os resultados obtidos com a aplicação dos algoritmos evolucionários aplicados ao problema do caixeiro viajante multiobjetivo. Foram utilizados algumas instâncias para avaliar o desempenho do MOGA (*Multiple Objective Genetic Algorithm*) e SPEA 2 (*Strenght Pareto Evolutionary Algorithm 2*).

Todas as instâncias utilizadas nos testes são completas e simétricas. Na Tabela 3 encontram-se as descrições dos problemas simétricos do caixeiro viajante utilizados neste trabalho.

Tabela 3: Problemas do Caixeiro Viajante utilizados.

| Instância | Nº Cidades | Descrição |
|---|---|---|
| brazil58 | 58 | Problema de 58 cidades no Brasil (TSPLIB) |
| sergipe24 | 24 | Problema de 24 cidades do Estado de Sergipe |
| brazil36 | 36 | Problema de 36 cidades do Brasil (DNIT) |

A ferramenta de desenvolvimento utilizada na implementação do MOGA e do SPEA foi o *Borland C++ Builder 6*. O *Borland C++ Builder* é um ambiente de desenvolvimento de aplicações orientadas a objetos que permite desenvolver software para o sistema operacional Windows, utilizando a linguagem de programação C++.

Os testes computacionais foram executados em um computador com processador Pentium 4 de 2 Ghz e 256 MB de memória RAM.

### 7.4.1 Parâmetros dos Algoritmos

A fim de comparar o desempenho dos dois algoritmos, todos os parâmetros foram definidos da mesma forma, especialmente o tamanho da população inicial. Na Tabela 4, mostramos os parâmetros utilizados. Os parâmetros foram definidos com base em estudos preliminares.

A implementação prevê um número máximo de 500 soluções aproximadas de Pareto a serem armazenadas.

Para cada instância foram realizados cinqüenta experimentos, para cada algoritmo, com sementes[5] aleatórias, obtendo o melhor valor dentre os experimentos e o pior valor.

Tabela 4: Parâmetros do MOGA e SPEA.

| MOGA | | | | SPEA | |
|---|---|---|---|---|---|
| **Gerações** | **População** | **Cruzamento** | **Mutação** | **Gerações** | **População** |
| 1000 | 250 | 0.8 | 0.1 | 10000 | 500 |

## *7.5* Análise dos Resultados

### 7.5.1 Instância Sergipe24

Esta instância foi criada a parti de 24 cidades do Estado de Sergipe e representa todas as possíveis conexões entre as 24 cidades, com suas respectivas distâncias geográficas.

Os resultados obtidos com esta instância estão exibidos na Tabela 5 e 6. O número de soluções encontradas e a comparação do desempenho dos melhores resultados obtidos pelos algoritmos, estão exibidos respectivamente nos gráficos das Figuras 32a, 32b, 32c, 32d, 32e, 32f, 32g, 32h, 32i, 32j e 32l.

Tabela 5: Comparação de Desempenho do PCV na Instância Sergipe24.

| **Algoritmo** | **Melhor Solução Encontrada (PCV/PML)** | **Pior Solução Encontrada (PCV/PML)** |
|---|---|---|
| MOGA | **1081** / 13112 | **1593** / 1438 |
| SPEA2 | **943** / 12355 | **976** / 11449 |

Tabela 6: Comparação de Desempenho do PML na Instância Sergipe24.

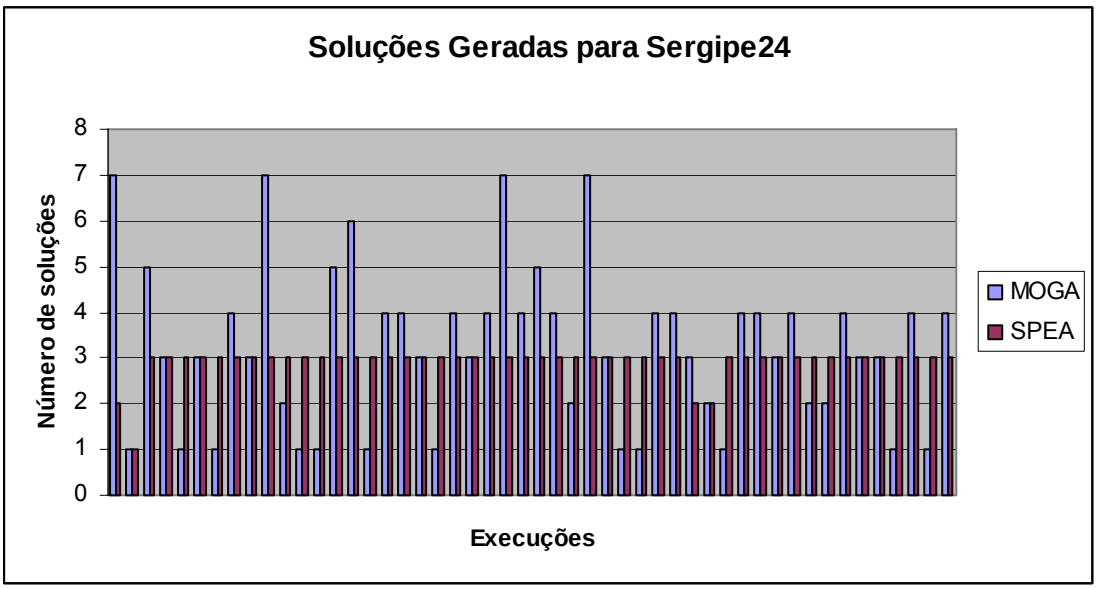
| **Algoritmo** | **Melhor Solução Encontrada (PCV/PML)** | **Pior Solução Encontrada (PCV/PML)** |
|---|---|---|
| MOGA | 1115 / **12170** | 1217 / **17492** |
| SPEA2 | 949 / **11114** | 951 / **12521** |

A Tabela 7 exibe o tempo médio computacional requerido pelo MOGA e SPEA nas 50 execuções.

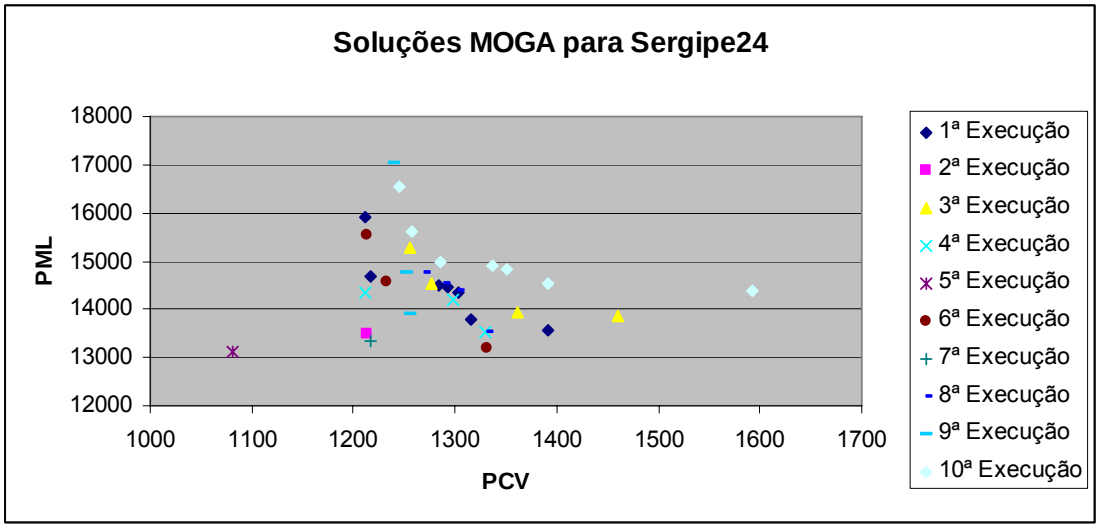
[5] Uso exclusivo no SPEA

Tabela 7: Tempo Médio Computacional para Instância Sergipe24.

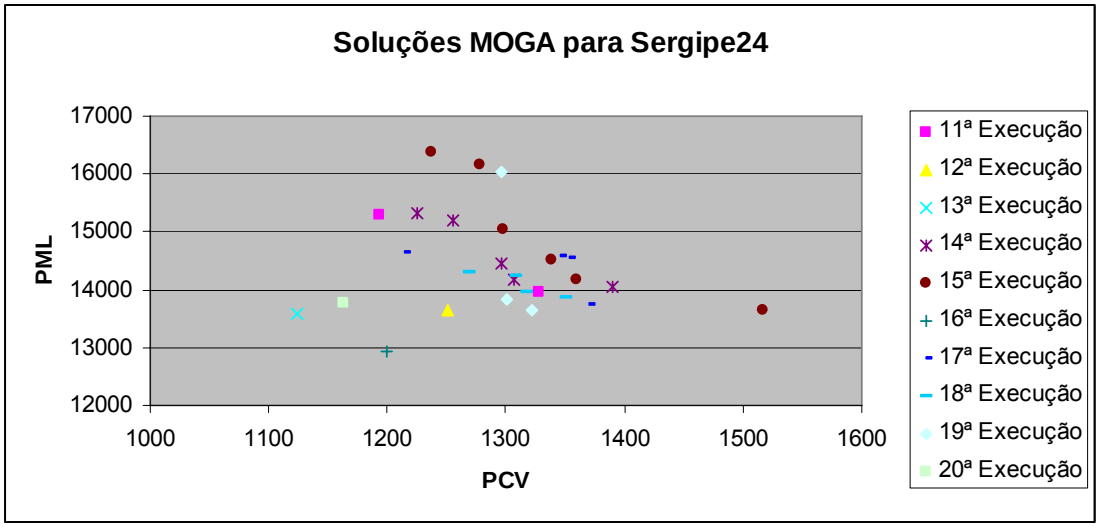
| Algoritmo | Tempo |
|---|---|
| MOGA | 37 s |
| SPEA | 159 s |



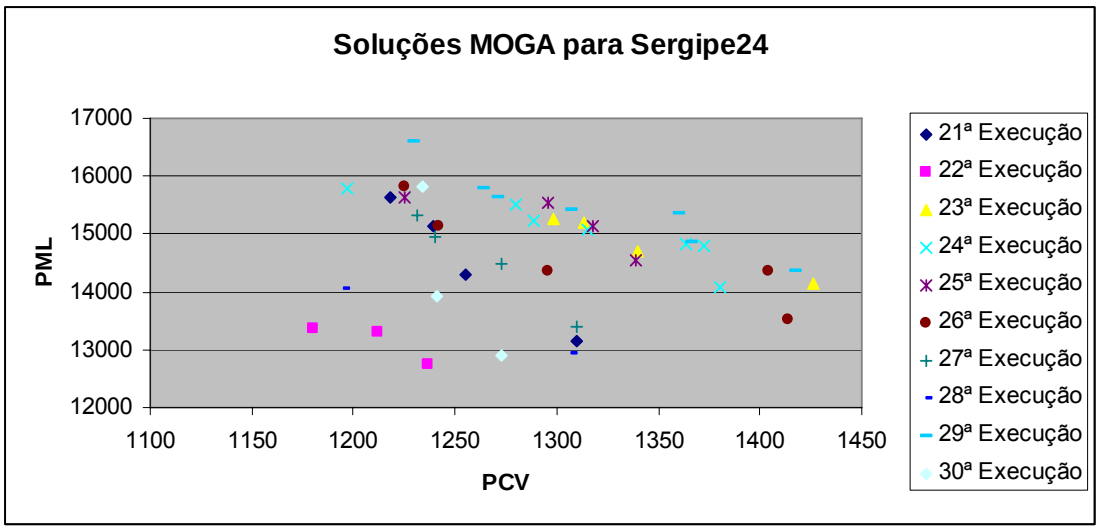

*a*



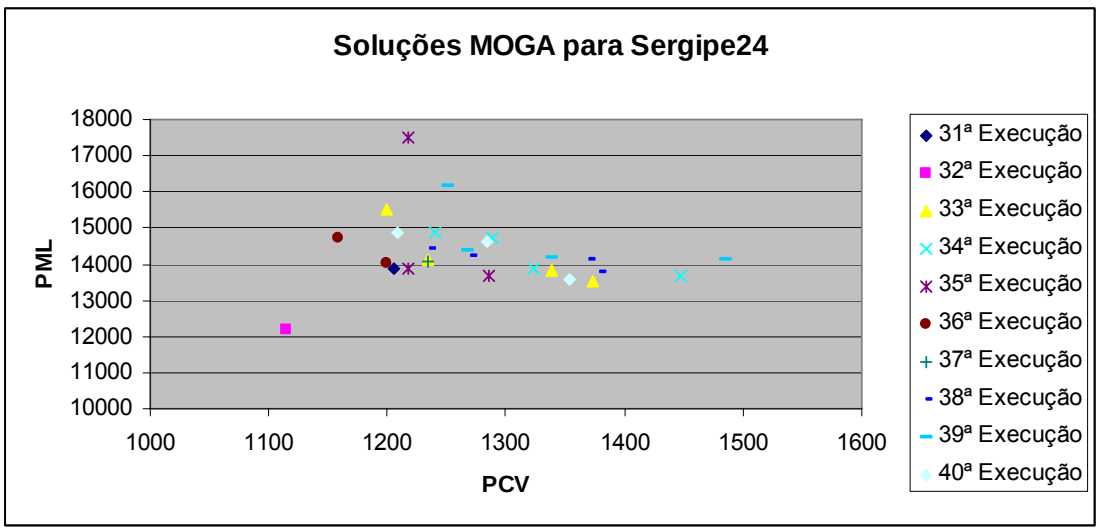

*b*


Soluções MOGA para Sergipe24
PML
PCV
17000
16000
15000
14000
13000
12000
1000
1100
1200
1300
1400
1500
1600
11ª Execução
12ª Execução
13ª Execução
14ª Execução
15ª Execução
16ª Execução
17ª Execução
18ª Execução
19ª Execução
20ª Execução

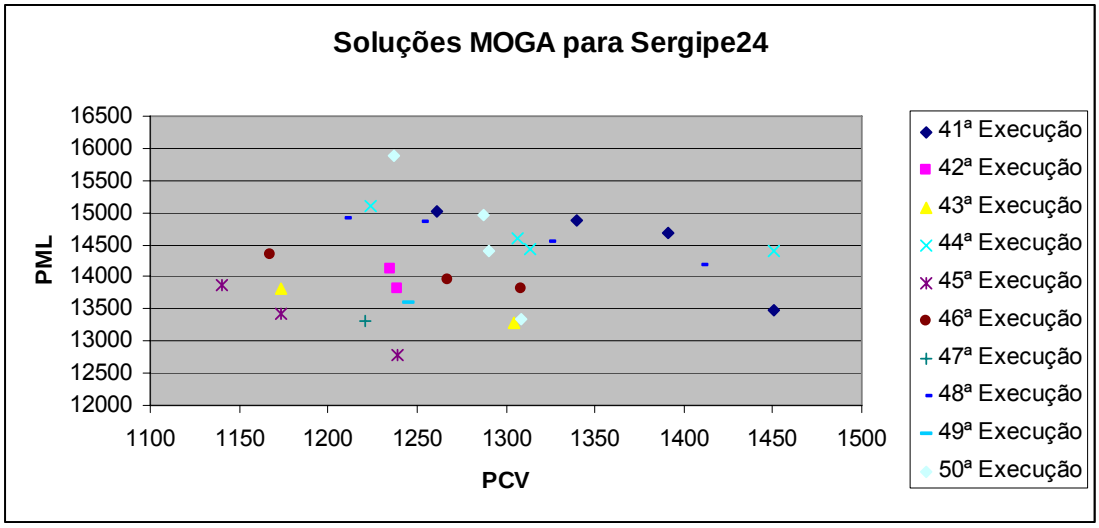
*c*


Soluções MOGA para Sergipe24
PML
PCV
17000
16000
15000
14000
13000
12000
1100
1150
1200
1250
1300
1350
1400
1450
21ª Execução
22ª Execução
23ª Execução
24ª Execução
25ª Execução
26ª Execução
27ª Execução
28ª Execução
29ª Execução
30ª Execução

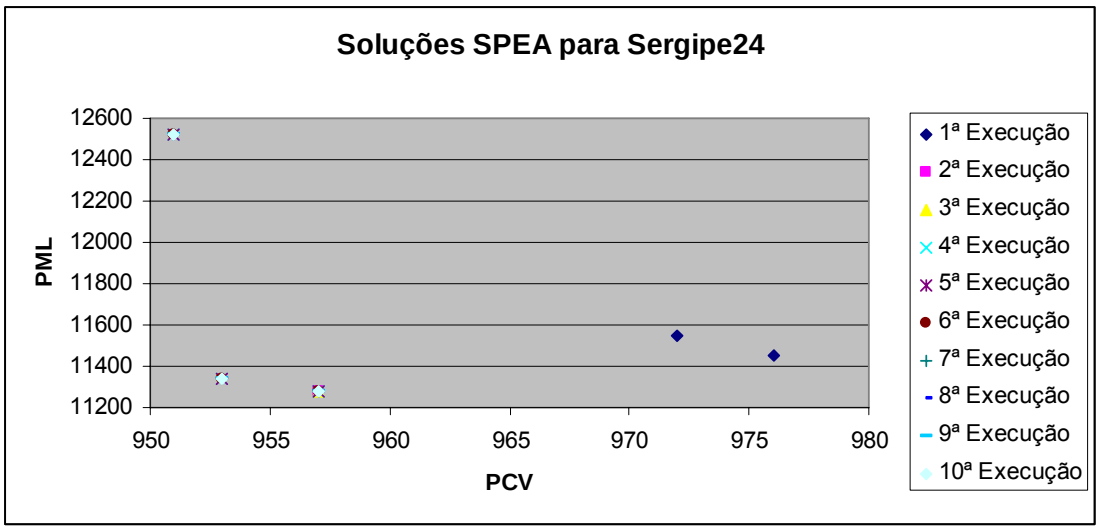
*d*


Soluções MOGA para Sergipe24
PML
PCV
18000
17000
16000
15000
14000
13000
12000
11000
10000
1000
1100
1200
1300
1400
1500
1600
31ª Execução
32ª Execução
33ª Execução
34ª Execução
35ª Execução
36ª Execução
37ª Execução
38ª Execução
39ª Execução
40ª Execução

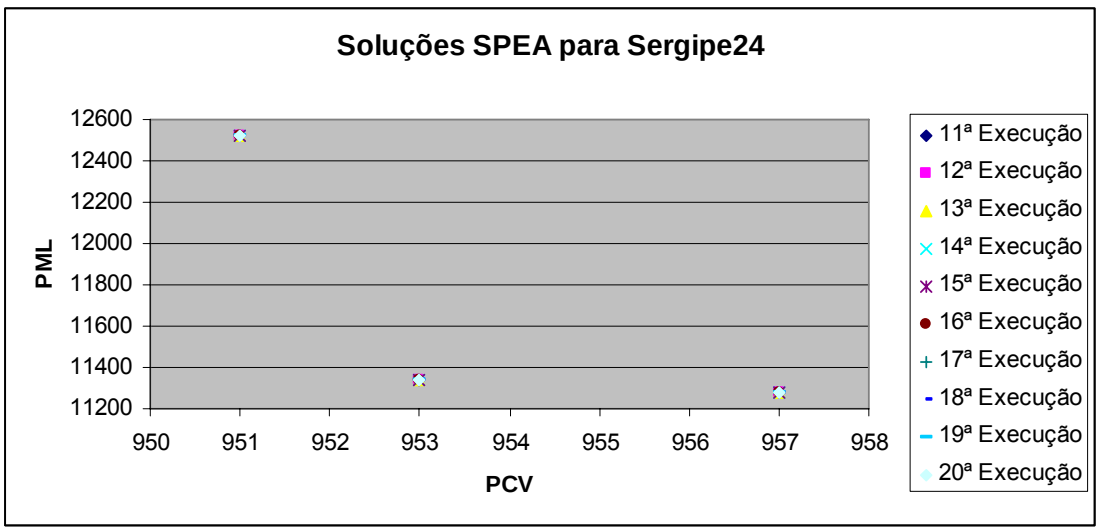
*e*


Soluções MOGA para Sergipe24
PML
16500
16000
15500
15000
14500
14000
13500
13000
12500
12000
1100
1150
1200
1250
1300
1350
1400
1450
1500
PCV
41ª Execução
42ª Execução
43ª Execução
44ª Execução
45ª Execução
46ª Execução
47ª Execução
48ª Execução
49ª Execução
50ª Execução

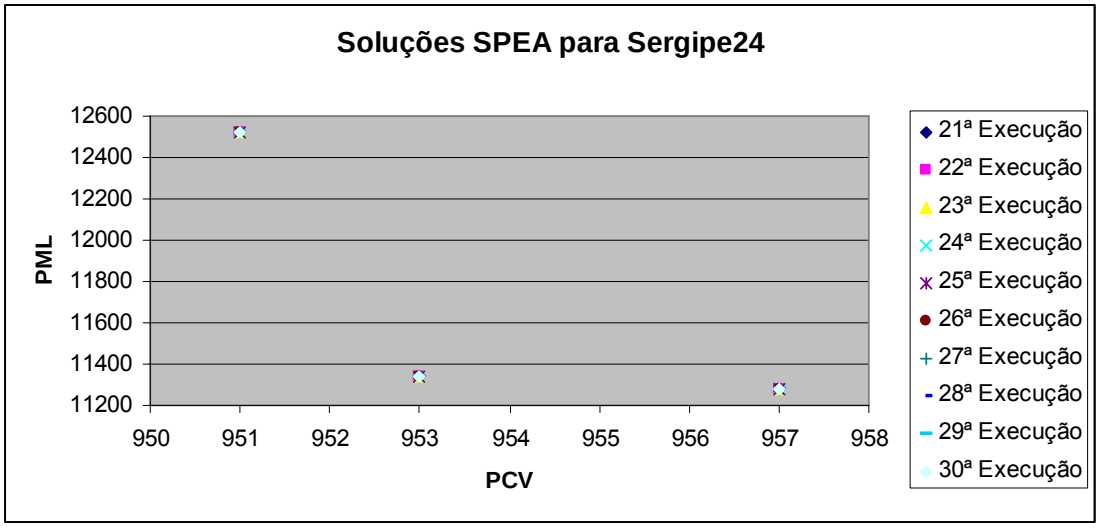
*f*


Soluções SPEA para Sergipe24
PML
12600
12400
12200
12000
11800
11600
11400
11200
950
955
960
965
970
975
980
PCV
1ª Execução
2ª Execução
3ª Execução
4ª Execução
5ª Execução
6ª Execução
7ª Execução
8ª Execução
9ª Execução
10ª Execução

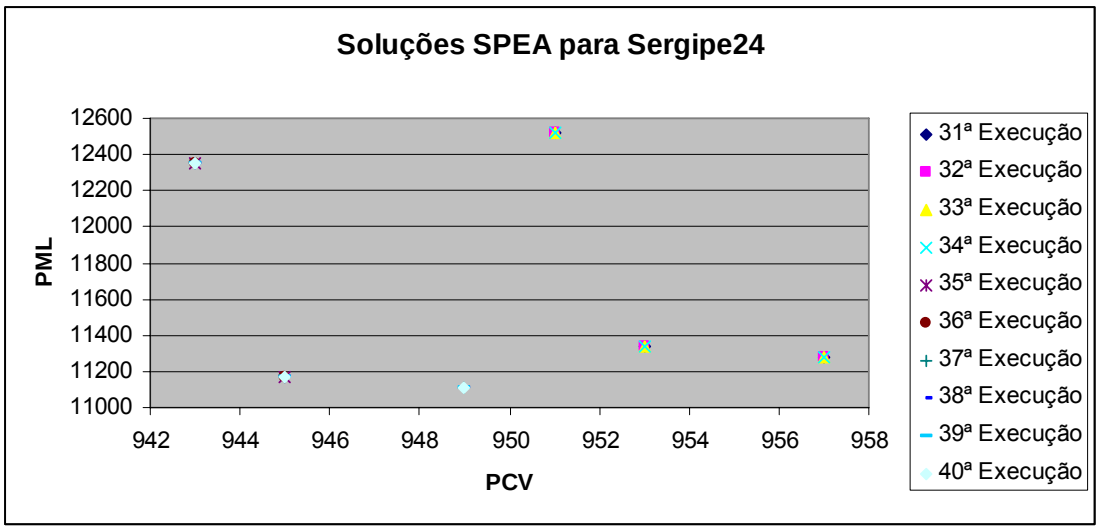
*g*


Soluções SPEA para Sergipe24
PML
12600
12400
12200
12000
11800
11600
11400
11200
950
951
952
953
954
955
956
957
958
PCV
11ª Execução
12ª Execução
13ª Execução
14ª Execução
15ª Execução
16ª Execução
17ª Execução
18ª Execução
19ª Execução
20ª Execução

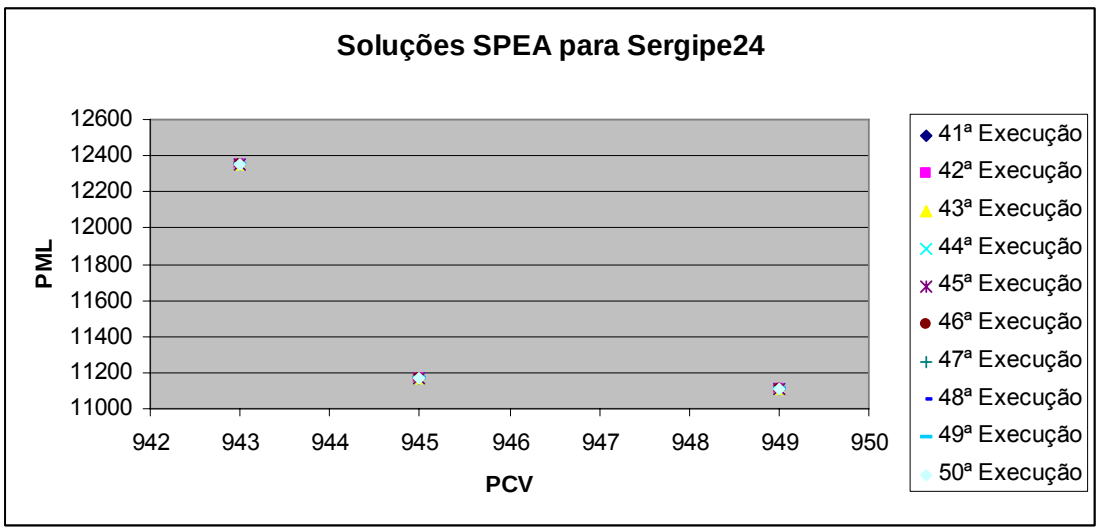
*h*




*i*




*j*




*l*

Figura 32: Análise Gráfica do MOGA e SPEA Aplicado à Instância Sergipe24.

### 7.5.2 Instância Brasil36

Esta instância foi retirada da base de dados do DNIT[6] e representa todas as possíveis conexões entre as 36 cidades do Brasil, com suas respectivas distâncias rodoviárias.

Os resultados obtidos com esta instância estão exibidos na Tabela 7 e 8. O número de soluções encontradas e a comparação do desempenho dos melhores resultados obtidos pelos algoritmos, estão exibidos respectivamente nos gráficos das Figuras 33a, 33b, 33c, 33d, 33e, 33f, 33g, 33h, 33i, 33j e 33l.

Tabela 8: Comparação de Desempenho do PCV na Instância Brasil36.

| Algoritmo | Melhor Solução Encontrada para o PCV (PCV/PML) | Pior Solução Encontrada para o PCV (PCV/PML) |
|---|---|---|
| MOGA | **32590** / 548650 | **48576** / 580820 |
| SPEA2 | **26471** / 381530 | **38329** / 538073 |

Tabela 9: Comparação de Desempenho do PML na Instância Brasil36.

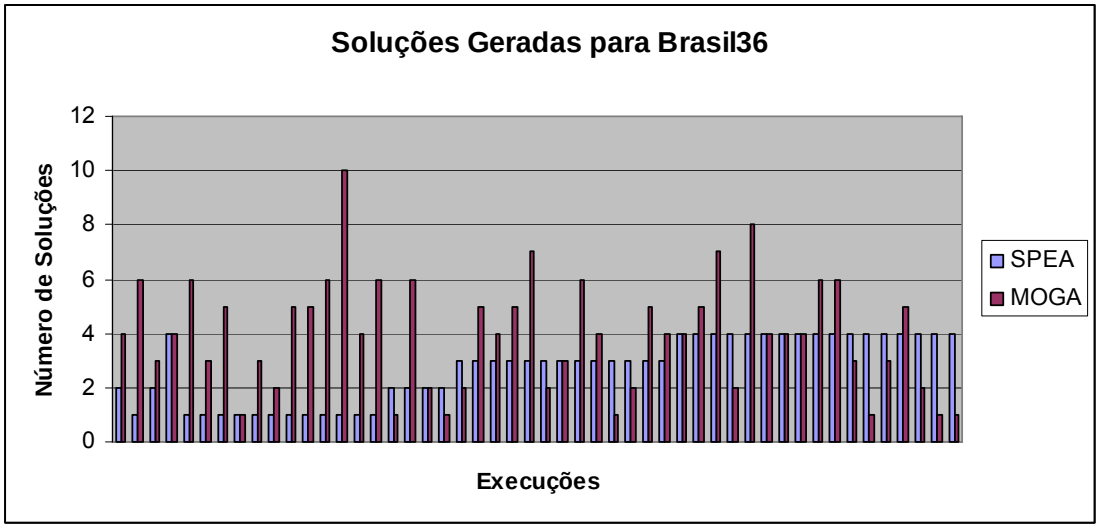
| Algoritmo | Melhor Solução Encontrada para o PML (PCV/PML) | Pior Solução Encontrada para o PML (PCV/PML) |
|---|---|---|
| MOGA | 37444 / **500197** | 37529 / **815997** |
| SPEA2 | 26530 / **378865** | 37995 / **550486** |

A Tabela 9 exibe o tempo médio computacional requerido pelo MOGA e SPEA nas 50 execuções.

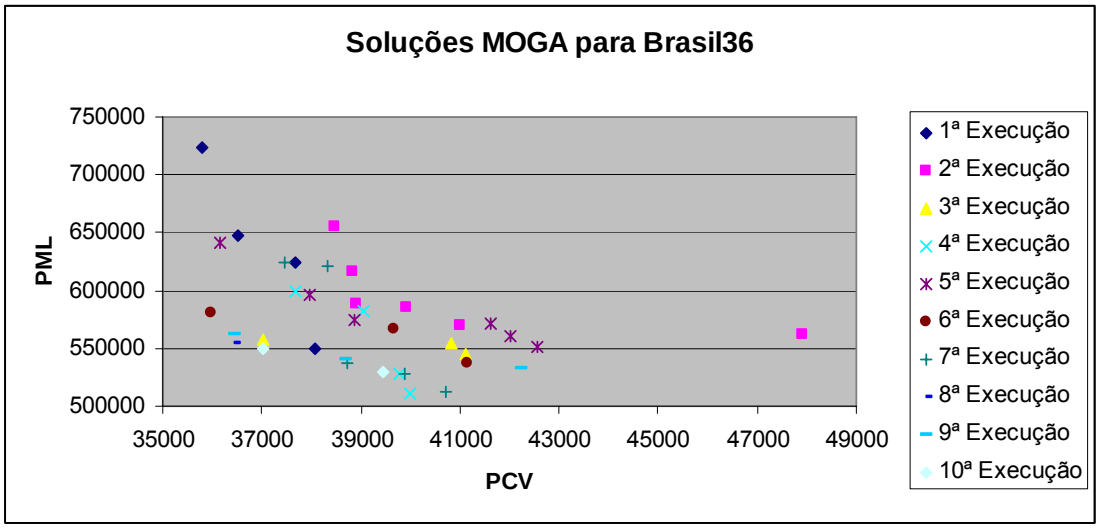
Tabela 10: Tempo Médio Computacional para Instância Brasil36.

| Algoritmo | Tempo |
|---|---|
| MOGA | 36 s |
| SPEA | 158 s |

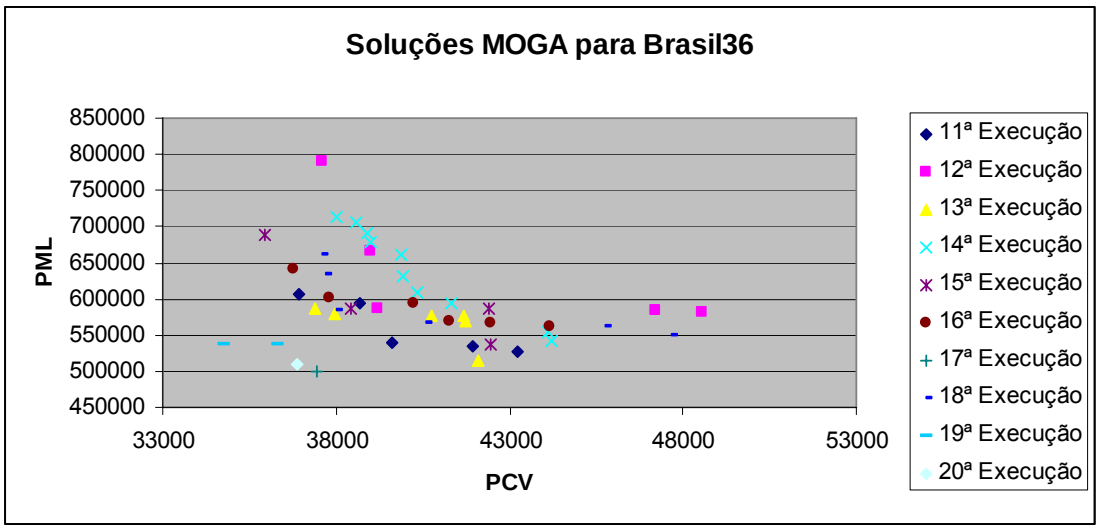
[6] Departamento Nacional de Infra-Estrutura de Transporte


Soluções Geradas para Brasil36
Número de Soluções
0
2
4
6
8
10
12
Execuções
SPEA
MOGA

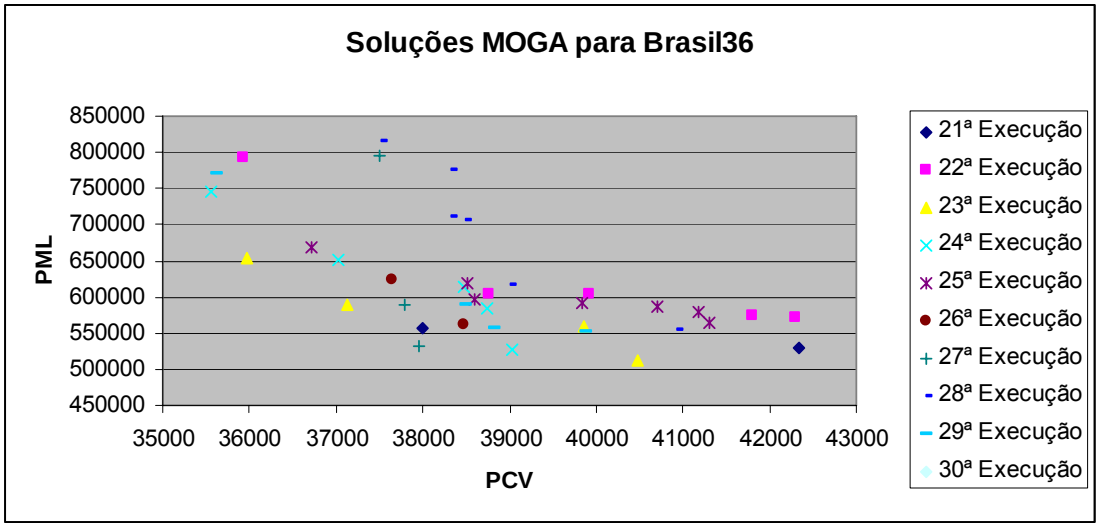
*a*


Soluções MOGA para Brasil36
PML
500000
550000
600000
650000
700000
750000
35000
37000
39000
41000
43000
45000
47000
49000
PCV
1ª Execução
2ª Execução
3ª Execução
4ª Execução
5ª Execução
6ª Execução
7ª Execução
8ª Execução
9ª Execução
10ª Execução

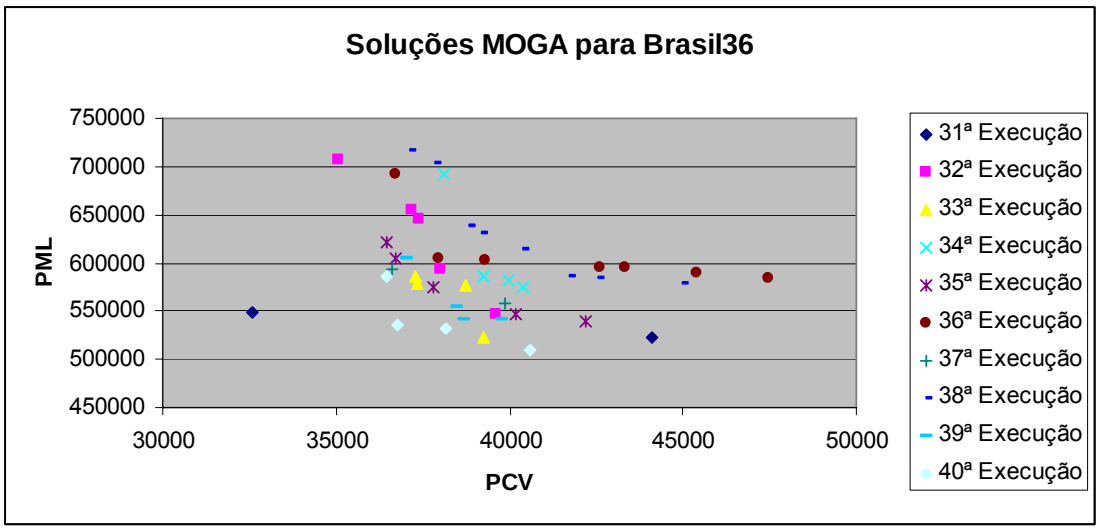
*b*


Soluções MOGA para Brasil36
PML
450000
500000
550000
600000
650000
700000
750000
800000
850000
33000
38000
43000
48000
53000
PCV
11ª Execução
12ª Execução
13ª Execução
14ª Execução
15ª Execução
16ª Execução
17ª Execução
18ª Execução
19ª Execução
20ª Execução

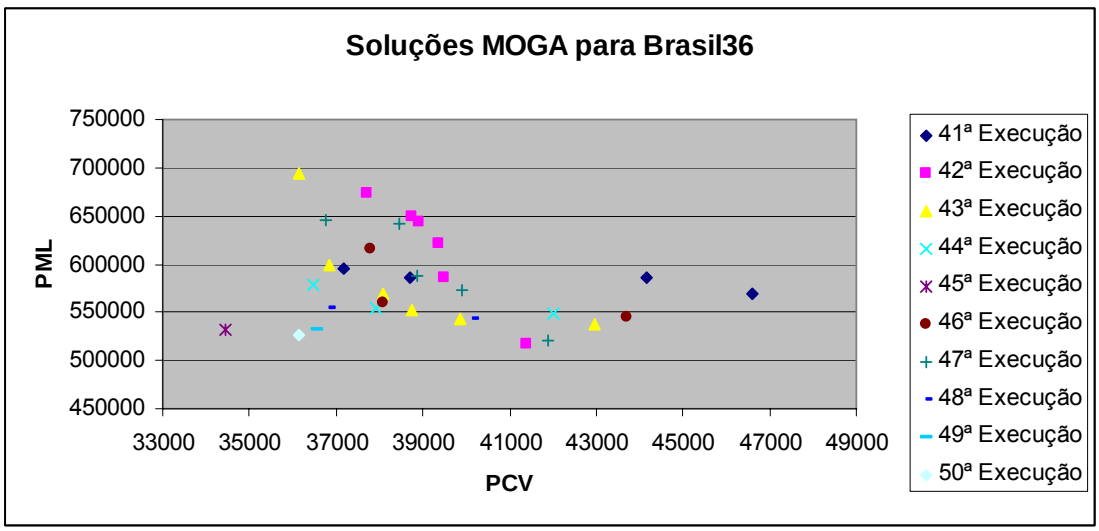
*c*


Soluções MOGA para Brasil36
PML
850000
800000
750000
700000
650000
600000
550000
500000
450000
35000
36000
37000
38000
39000
40000
41000
42000
43000
PCV
21ª Execução
22ª Execução
23ª Execução
24ª Execução
25ª Execução
26ª Execução
27ª Execução
28ª Execução
29ª Execução
30ª Execução

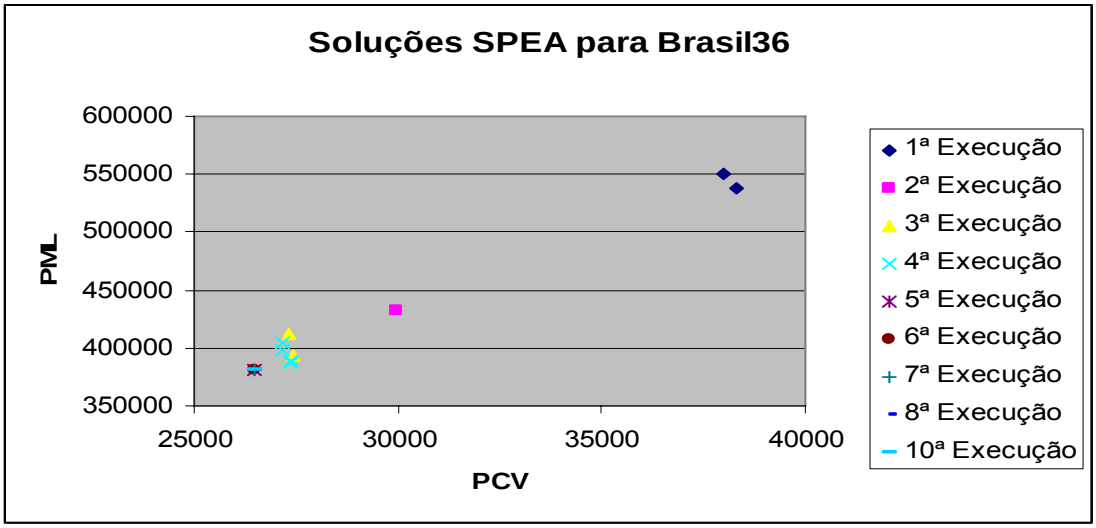
*d*


Soluções MOGA para Brasil36
PML
750000
700000
650000
600000
550000
500000
450000
30000
35000
40000
45000
50000
PCV
31ª Execução
32ª Execução
33ª Execução
34ª Execução
35ª Execução
36ª Execução
37ª Execução
38ª Execução
39ª Execução
40ª Execução

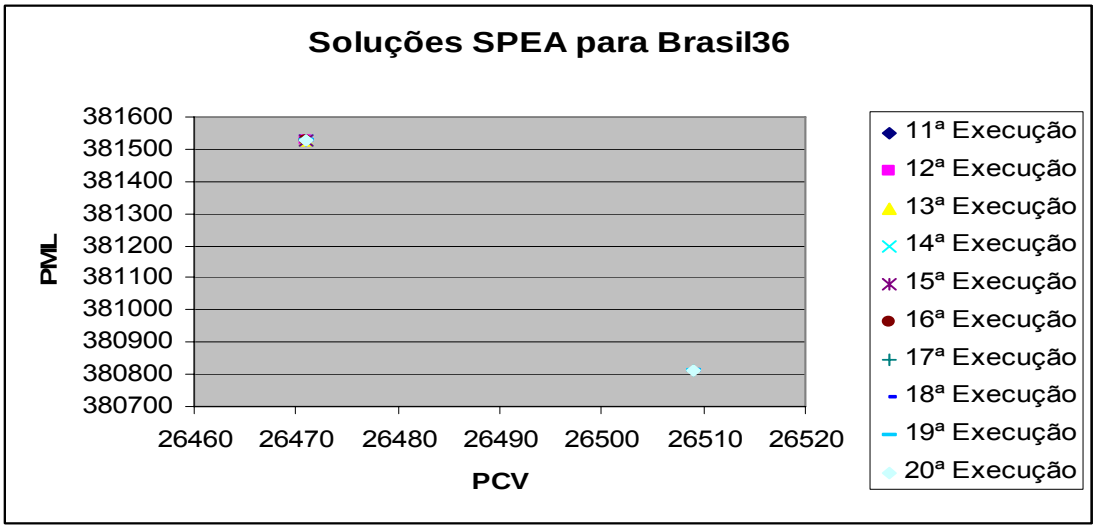
*e*


Soluções MOGA para Brasil36
PML
750000
700000
650000
600000
550000
500000
450000
33000
35000
37000
39000
41000
43000
45000
47000
49000
PCV
41ª Execução
42ª Execução
43ª Execução
44ª Execução
45ª Execução
46ª Execução
47ª Execução
48ª Execução
49ª Execução
50ª Execução

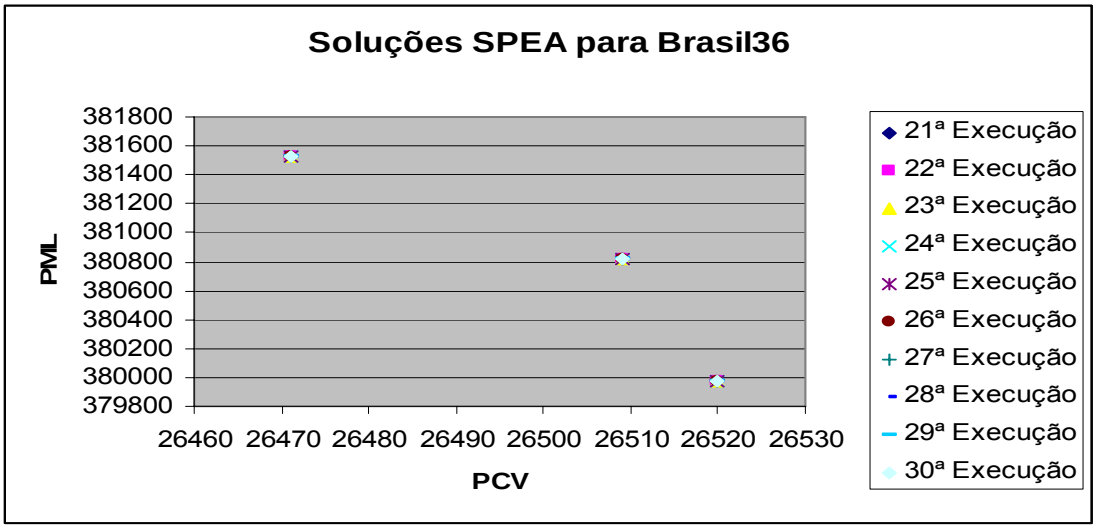
*f*


Soluções SPEA para Brasil36
PML
600000
550000
500000
450000
400000
350000
25000
30000
35000
40000
PCV
1ª Execução
2ª Execução
3ª Execução
4ª Execução
5ª Execução
6ª Execução
7ª Execução
8ª Execução
10ª Execução

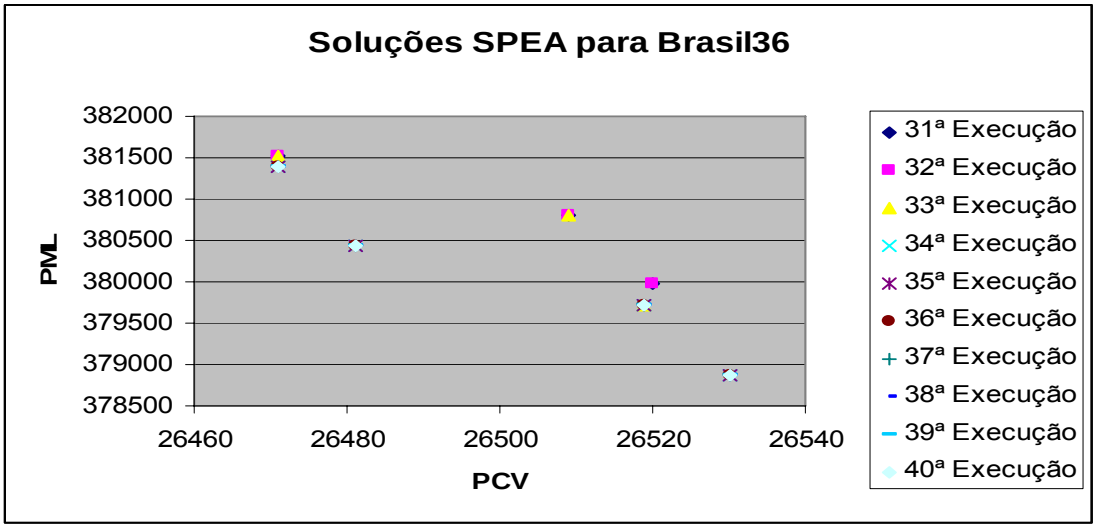
*g*


Soluções SPEA para Brasil36
PML
381600
381500
381400
381300
381200
381100
381000
380900
380800
380700
26460
26470
26480
26490
26500
26510
26520
PCV
11ª Execução
12ª Execução
13ª Execução
14ª Execução
15ª Execução
16ª Execução
17ª Execução
18ª Execução
19ª Execução
20ª Execução

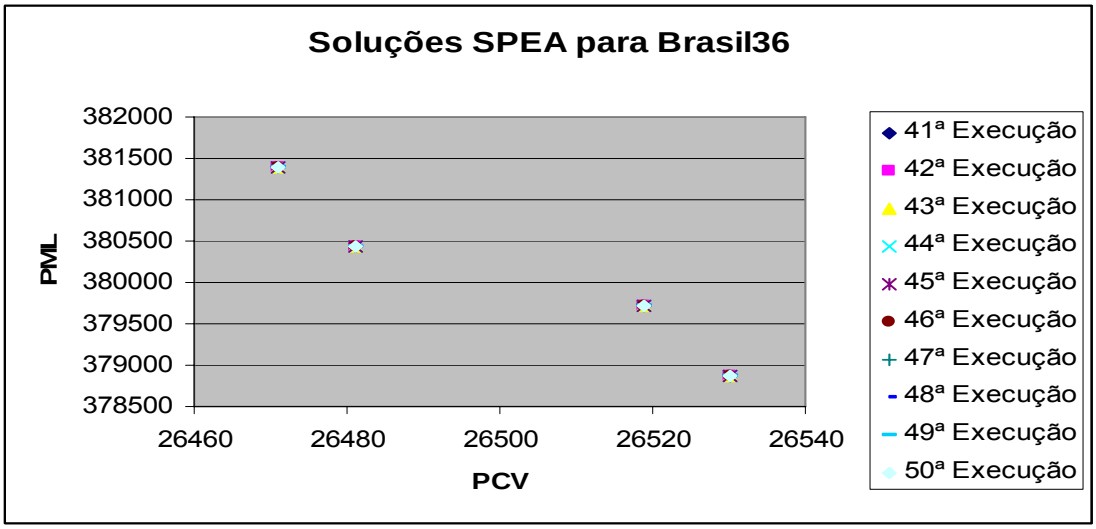
*h*


Soluções SPEA para Brasil36
PML
381800
381600
381400
381200
381000
380800
380600
380400
380200
380000
379800
26460
26470
26480
26490
26500
26510
26520
26530
PCV
21ª Execução
22ª Execução
23ª Execução
24ª Execução
25ª Execução
26ª Execução
27ª Execução
28ª Execução
29ª Execução
30ª Execução

*i*




*j*




*l*

Figura 33: Análise Gráfica do MOGA e SPEA Aplicado à Instância Brasil36.

### 7.5.3 Instância Brazil58

Esta instância foi retirada da base de dados do TSPLIB e representam todas as possíveis conexões entre as 58 cidades no Brasil, com suas respectivas distâncias geográficas.

Os resultados obtidos com esta instância estão exibidos na Tabela 10 e 11. Em parênteses, encontra-se o erro encontrado por cada algoritmo. O número de soluções encontradas e a comparação do desempenho dos melhores resultados obtidos pelos algoritmos, estão exibidos respectivamente nos gráficos das Figuras 34a, 34b, 34c, 34d, 34e, 34f, 34g, 34h, 34i, 34j e 34l.

Tabela 11: Comparação de Desempenho do PCV na Instância Brazil58.

| Algoritmo | Melhor sol. PCV | Melhor Solução Encontrada para o PCV (PCV/PML) | Pior Solução Encontrada para o PCV (PCV/PML) |
|---|---|---|---|
| MOGA | 25395 | **60969** (140%) / 1,78E+11 | **87030** (242,70%) / 1,89E+11 |
| SPEA2 | | **25420** (0,098%) / 642785 | **36637** (44,26%) / 881742 |

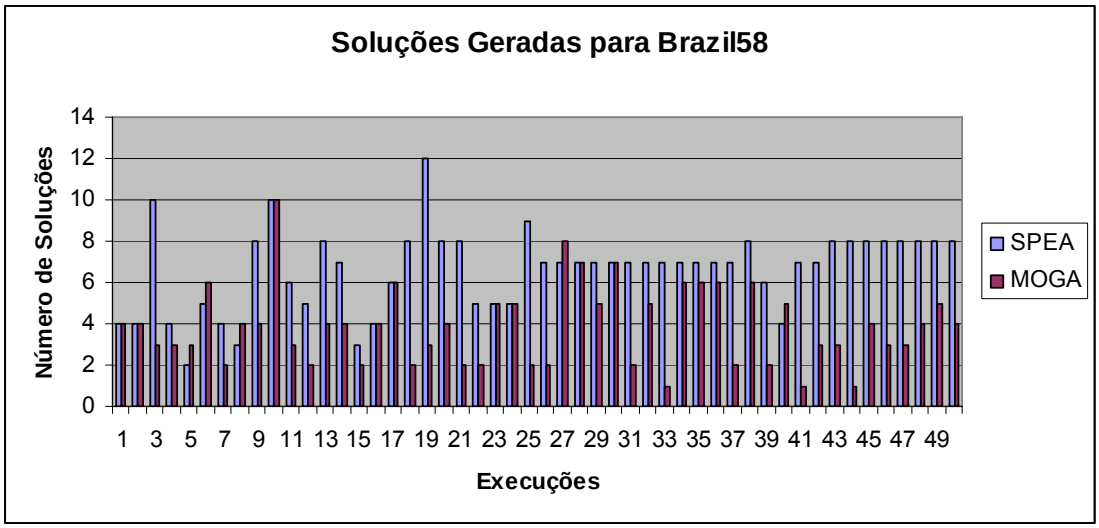
Tabela 12: Comparação de Desempenho do PML na Instância Brazil58.

| Algoritmo | Melhor Solução Encontrada para o PML (PCV/PML) | Pior Solução Encontrada para o PML (PCV/PML) |
|---|---|---|
| MOGA | 80514 / **1,85E+09** | 72626 / **2,44E+11** |
| SPEA2 | 29210 / **607890** | 34018 / **995183** |

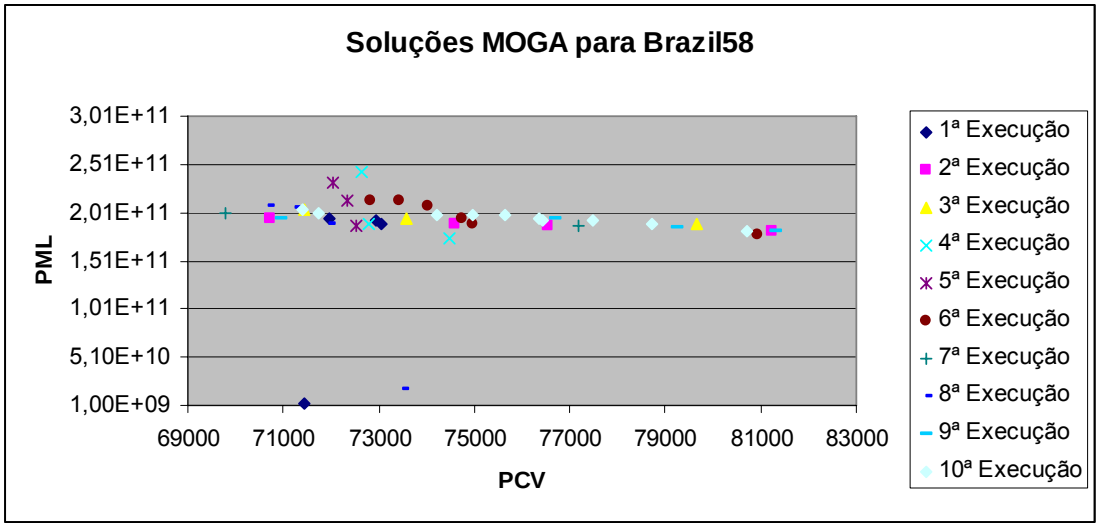
A Tabela 12 exibe o tempo médio computacional requerido pelo MOGA e SPEA nas 50 execuções.

Tabela 13: Tempo Médio Computacional para Instância Brasil36.

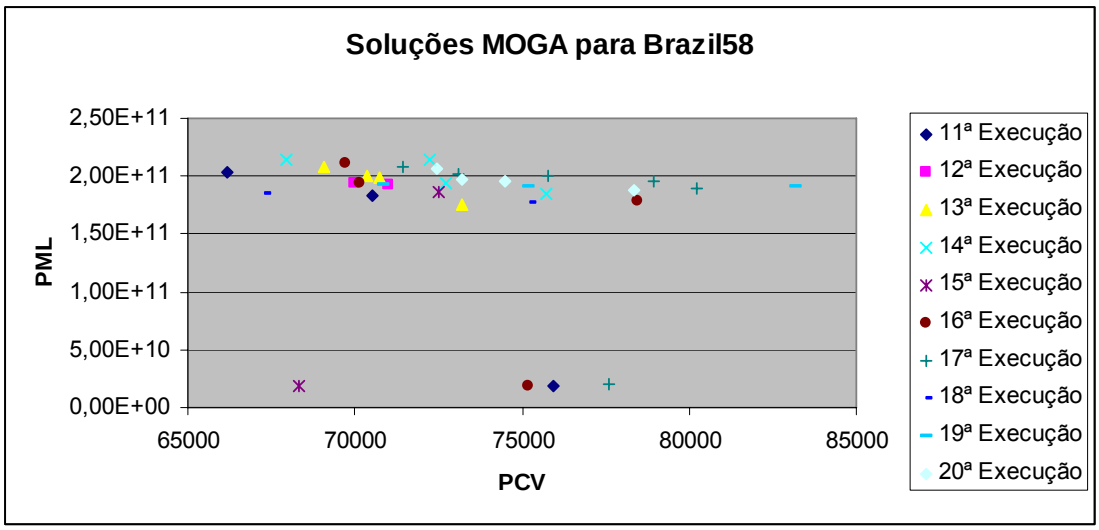
| Algoritmo | Tempo |
|---|---|
| MOGA | 37 s |
| SPEA | 335 s |



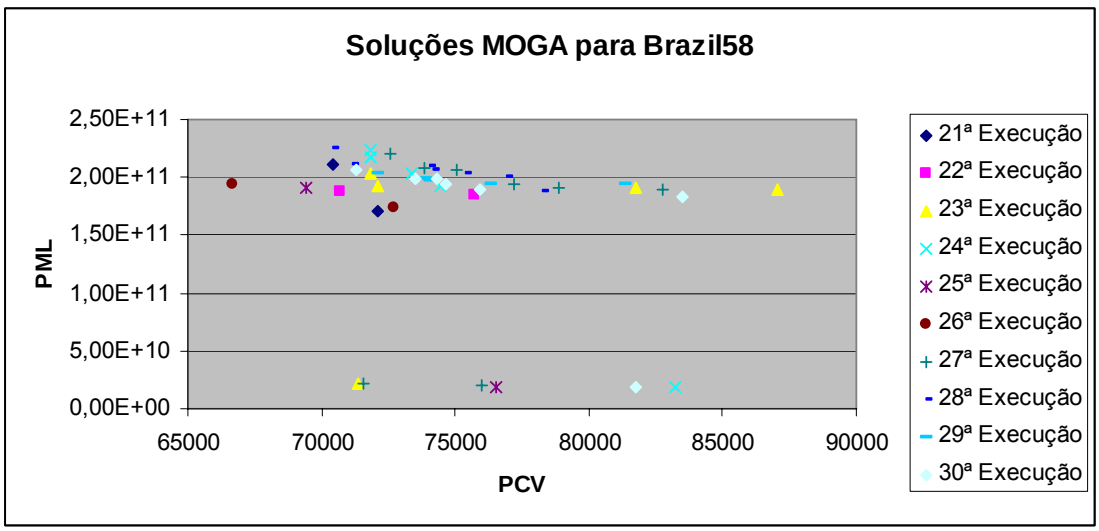

*a*


Soluções MOGA para Brazil58
PML
3,01E+11
2,51E+11
2,01E+11
1,51E+11
1,01E+11
5,10E+10
1,00E+09
69000
71000
73000
75000
77000
79000
81000
83000
PCV
1ª Execução
2ª Execução
3ª Execução
4ª Execução
5ª Execução
6ª Execução
7ª Execução
8ª Execução
9ª Execução
10ª Execução

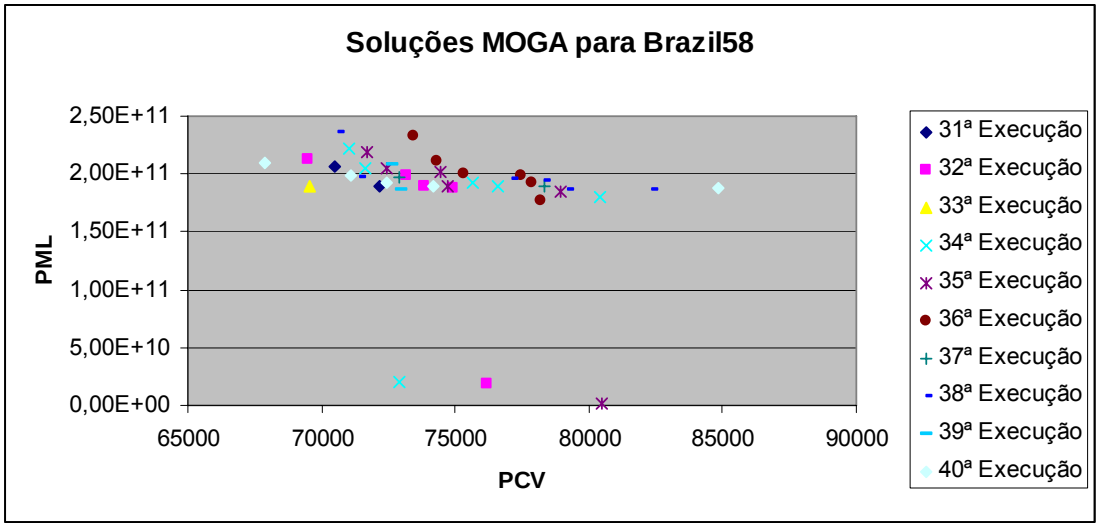
*b*


Soluções MOGA para Brazil58
PML
2,50E+11
2,00E+11
1,50E+11
1,00E+11
5,00E+10
0,00E+00
65000
70000
75000
80000
85000
PCV
11ª Execução
12ª Execução
13ª Execução
14ª Execução
15ª Execução
16ª Execução
17ª Execução
18ª Execução
19ª Execução
20ª Execução

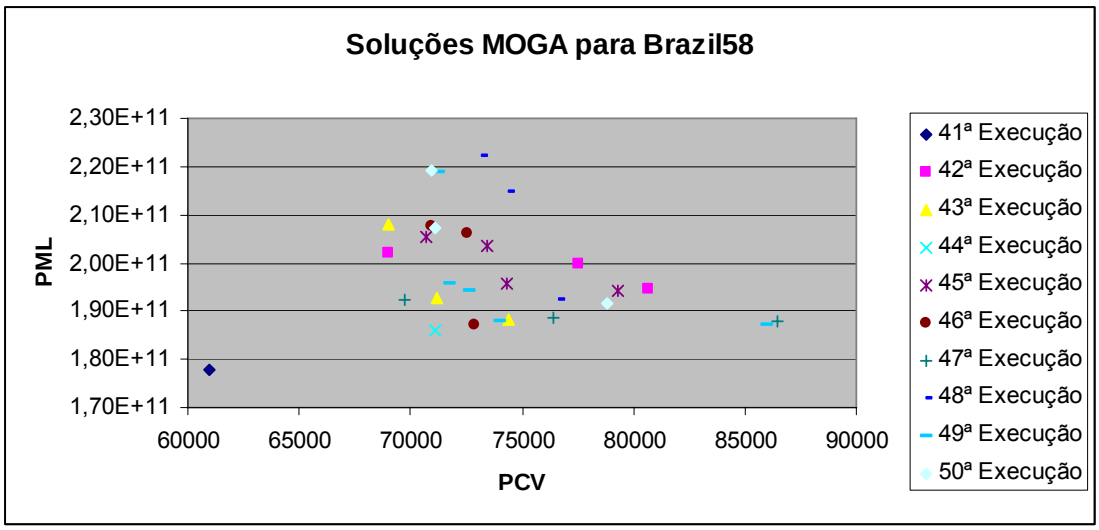
c


Soluções MOGA para Brazil58
PML
2,50E+11
2,00E+11
1,50E+11
1,00E+11
5,00E+10
0,00E+00
65000
70000
75000
80000
85000
90000
PCV
21ª Execução
22ª Execução
23ª Execução
24ª Execução
25ª Execução
26ª Execução
27ª Execução
28ª Execução
29ª Execução
30ª Execução

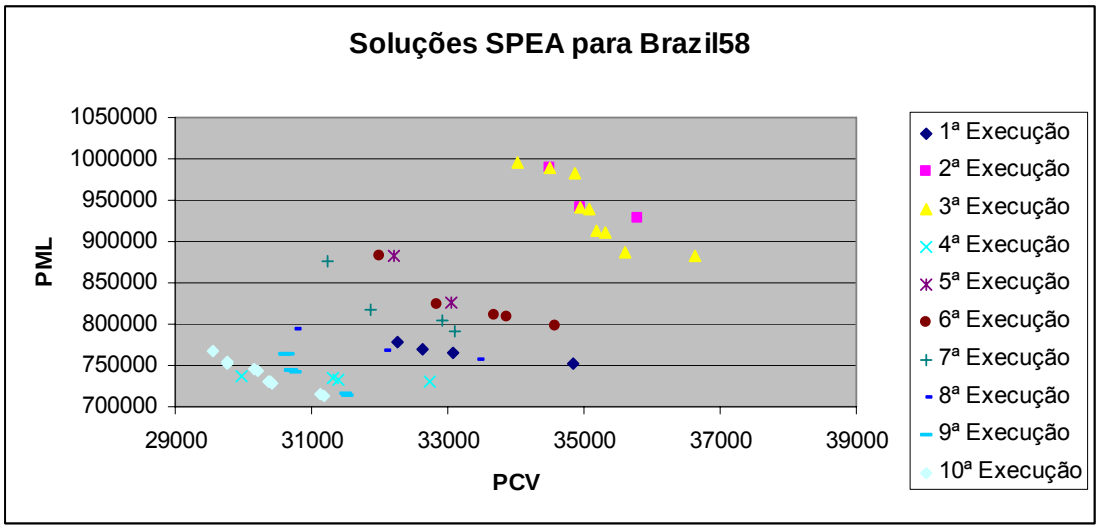
d


Soluções MOGA para Brazil58
2,50E+11
2,00E+11
1,50E+11
1,00E+11
5,00E+10
0,00E+00
PML
65000
70000
75000
80000
85000
90000
PCV
31ª Execução
32ª Execução
33ª Execução
34ª Execução
35ª Execução
36ª Execução
37ª Execução
38ª Execução
39ª Execução
40ª Execução

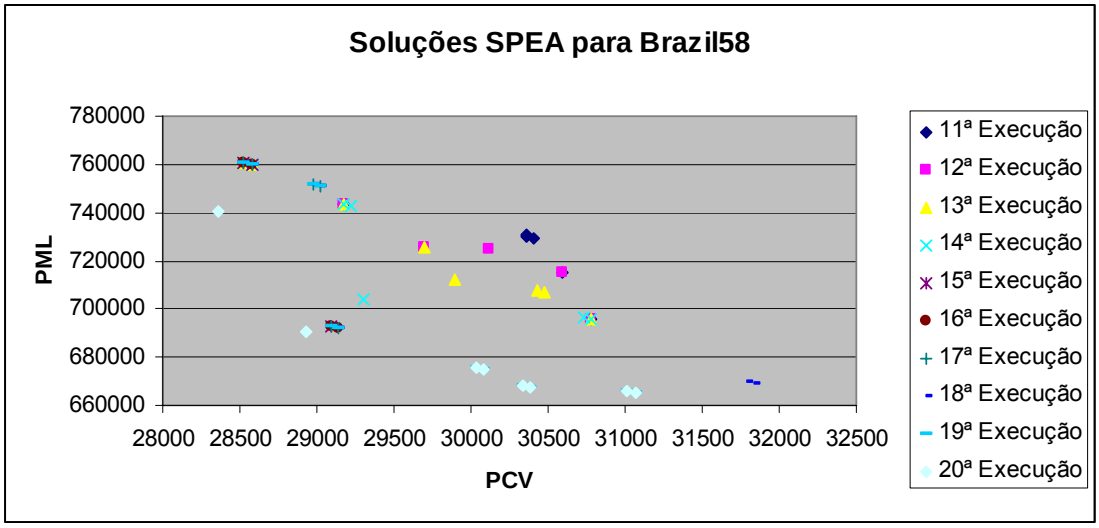
e


Soluções MOGA para Brazil58
2,30E+11
2,20E+11
2,10E+11
2,00E+11
1,90E+11
1,80E+11
1,70E+11
PML
60000
65000
70000
75000
80000
85000
90000
PCV
41ª Execução
42ª Execução
43ª Execução
44ª Execução
45ª Execução
46ª Execução
47ª Execução
48ª Execução
49ª Execução
50ª Execução

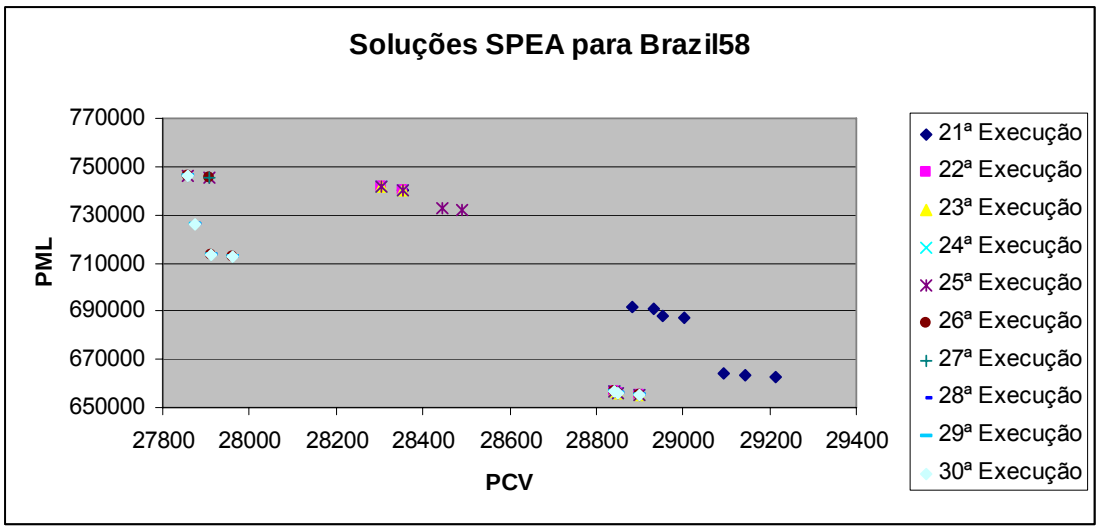
f


Soluções SPEA para Brazil58
1050000
1000000
950000
900000
850000
800000
750000
700000
PML
29000
31000
33000
35000
37000
39000
PCV
1ª Execução
2ª Execução
3ª Execução
4ª Execução
5ª Execução
6ª Execução
7ª Execução
8ª Execução
9ª Execução
10ª Execução

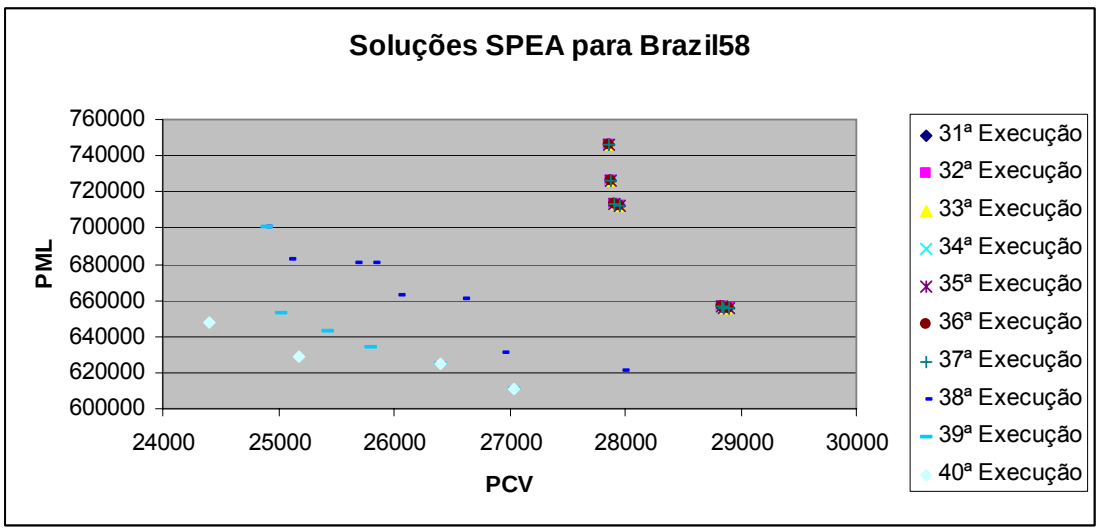
g


Soluções SPEA para Brazil58
780000
760000
740000
720000
700000
680000
660000
PML
28000 28500 29000 29500 30000 30500 31000 31500 32000 32500
PCV
11ª Execução
12ª Execução
13ª Execução
14ª Execução
15ª Execução
16ª Execução
17ª Execução
18ª Execução
19ª Execução
20ª Execução

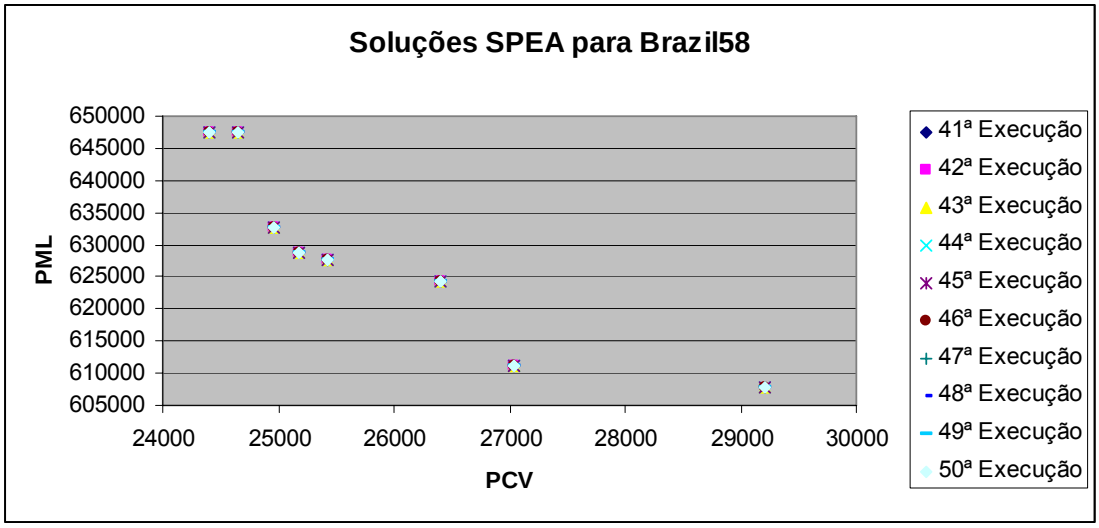
h


Soluções SPEA para Brazil58
770000
750000
730000
710000
690000
670000
650000
PML
27800 28000 28200 28400 28600 28800 29000 29200 29400
PCV
21ª Execução
22ª Execução
23ª Execução
24ª Execução
25ª Execução
26ª Execução
27ª Execução
28ª Execução
29ª Execução
30ª Execução

i


Soluções SPEA para Brazil58
760000
740000
720000
700000
680000
660000
640000
620000
600000
PML
24000 25000 26000 27000 28000 29000 30000
PCV
31ª Execução
32ª Execução
33ª Execução
34ª Execução
35ª Execução
36ª Execução
37ª Execução
38ª Execução
39ª Execução
40ª Execução

j




/

Figura 34: Análise Gráfica do MOGA e SPEA Aplicado à Instância Brazil58.

Tanto o algoritmo MOGA quanto o SPEA2 conseguiram encontrar soluções possivelmente não-dominadas. Entretanto, o algoritmo SPEA2 obteve um desempenho melhor que o MOGA, pois a melhor solução encontrada para o PCV está próxima da solução ótima conhecida.

Esta diferencia nos resultados dos algoritmos pode ser explicada pelo fato do algoritmo SPEA2 utilizar o elitismo através de uma população.

# CAPÍTULO 8

# CONCLUSÕES E TRABALHOS FUTUROS

O objetivo deste trabalho foi desenvolver algoritmos evolucionários para resolver o problema do caixeiro viajante multiobjetivo.

Primeiramente, foi desenvolvido o *Multiple Objective Genetic Algorithm* (MOGA) para obter um conjunto de soluções Pareto ótimas para o problema. Foi considerada a minimização da mínima latência e da distância entre as cidades. Os resultados computacionais mostram que o MOGA é, razoavelmente, rápido e gera uma aproximação razoável do conjunto de Pareto ótimo.

Além disso, foi desenvolvido o *Strenght Pareto Evolutionary Algorithm 2* (SPEA 2) para resolver o mesmo problema. Esse algoritmo foi testado e apresentou bom desempenho quando comparado com o MOGA.

De modo geral, esse trabalho contribuiu com algumas idéias e técnicas a serem usadas em problemas de otimização combinatória multiobjetivo. Dentre as técnicas apresentadas, podemos destacar os algoritmos evolucionários, como uma técnica que permite a interferência do tomador de decisão depois do processo de solução. O uso de algoritmos evolucionários possibilita explorar várias soluções em paralelo, gerando rapidamente várias soluções dominantes distribuídas e bem próximas às soluções Pareto ótimo.

Como sugestão de trabalhos futuros, podemos citar:

- Otimização de três ou mais objetivos para o problema do caixeiro viajante multiobjetivo;

- Aplicação de algoritmos evolucionários para resolver outros problemas de otimização combinatória multiobjetivo;
- Comparação de algoritmos evolucionários com outras técnicas de solução, tipo: Multi-objetive *Simulated Annealing* (MOSA) e *Pareto Simulated Annealing* (PSA);
- Comparação com soluções ótimas para cada critério, avaliando a degradação da qualidade de solução com respeito a cada critérios isoladamente.

# REFERÊNCIAS BIBLIOGRÁFICAS


Agarwala,R., Applegate,D., Maglott, D., Schuler, G., Schaffler, A. (2000). **A Fast and scalable radiation hybrid map construction and integration strategy**. Genome Research 10; 350-364.

Andersson, J., Pohl J. and Krus, P. (1998). ***Design of objective functions for optimization of multi-domain systems***, Proceedings of the 1998 ASME Winter Meeting, Fuild Power Systems and Technology, Anaheim, California, November 15-20.

Andersson, J. (2000) ***A survey of multiobjective optimization in engineering design***, Technical Report No. LiTH-IKP-R-1097, Department of Mechanical Engineering, Linköping University.

Arroyo, J. E. C. (2002) ***Heuristicas e metaheuristicas para otimização combinatória multiobjetivo***, Tese, Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas-SP.

Arroyo, J. E. C. e Armentano, V. A. (2003) ***Algoritmo genético para o problema do caixeiro viajante multiobjetivo***, IV Encontro Nacional de Inteligência Artificial, Campinas, SP, Brasil.

Kennedy J. and Eberhart R. (1995) ***Particle swarm optimization***, Proc. IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, J,pp. IV: 1942-1948.

Avner, P., Bruls, T., Poras, I., Eley, L., Gas, S., Ruiz, P., Wiles, M., Sousa-Nunes, R., Kettleborough, R., Rana, A., Morissette, J., Bentley, L., Goldsworthy, M., Haynes, A., Herbert, E., Southam, L., Lehrach, H., Weissenbach, J., Manenti, G., Rodriguez-Tome, P., Beddingtonll, R., Dunwoodie, S., Cox, R.

(2001) **A radiation hybrid transcript may of the mouse genome**, Nature Genetics 29, 194 – 200.

Benayoun R., de Montgolfier J., Tergny J., and Laritchev O. (1971) ***Linear programming with multiple objective functions: step method (STEM)***, Mathematical Programming, vol. 1, pp. 366-375.

Bentley, J.L. (1992). ***Fast algorithms for geometric traveling salesman problems***. ORSA Journal on Computing, 4(4):387–411.

Biggs, Norman L., Lloyd, E. Keith, and Wilson, Robin J. (1976) **Graph theory 1736-1936.** Oxford University Press, London.

Borges, P. C., Hansen, M. P. (2002) ***A Study of Global Convexity for a Multiple Objective Traveling Salesman Problem***. In Ribeiro, C. C., Hansen, P. (eds.): Essays and Surveys in Metaheuristics. Kluwer Academic Publishers Norwell, MA 129-150.

Campello R. E., Ampello, R. E. e Maculan, N. (1994) ***Algoritmo e heurísticas desenvolvimento e avaliação de performance***. Editora da Universidade Federal Fluminense, Niterói-RJ.

Castro, R. E. (2001) ***Otimização de estruturas com multi-objetivo via algoritmos genéticos***, Tese, Universidade Federal do Rio de Janeiro, Rio de Janeiro-RJ.

Charnes A., Cooper W. W., and Ferguson R. O. (1955), ***Optimal estimation of executive compensation by linear programing***, Management Science, vol. 1, pp. 138-151.

Chiampi M., G F., Ch M., C R., and M. R. (1998), ***Multi-objective optimization with stochastic algorithms and fuzzy definition of objective function***, International Journal of Applied Electromagnetics in Materials, vol. 9, pp. 381-389.

Cochran, Jeffery K. Horng, Shwu-Min and Fowler, John W. (2003). ***A multipopulation genetic algorithm to solve multi-objective scheduling problems for parallel machines***. Comput. Oper. Res., 30(7):1087-1102.

Coello, C. A. C. (1998) ***An updated survey of GA-based multiobjective optimization techniques***, Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, México.

Coello C. A. C. (2001) ***A short tutorial on evolutionary multiobjective optimization***. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, First International Conference on Evolutionary Multi-Criterion Optimization, pages 21-40. Springer-Verlag. Lecture Notes in Computer Science No. 1993.

Coello, Carlos A. Coello and Pulido, Gregorio Toscano (2001). ***A Micro-Genetic Algorithm for Multiobjective Optimization.*** In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, First International Conference on Evolutionary Multi-Criterion Optimization, pages 126-140. Springer-Verlag. Lecture Notes in Computer Science No. 1993.

Coello, Carlos A. Coello and Becerra, Ricardo Landa (2003). ***Evolutionary Multiobjective Optimization using a Cultural Algorithm***. In 2003 IEEE Swarm Intelligence Symposium Proceedings, pages 6-13, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

Coello, Carlos A. Coello, Pulido, Gregorio Toscano and Lechuga, Maximino Salazar (2004). ***Handling Multiple Objectives With Particle Swarm Optimization***. IEEE Transactions on Evolutionary Computation, 8(3):256-279, June 2004.

Cohon J.L. (1978) ***Multiobjective programing and plannig***. New York: Academic Press.

Conway, R., Maxwell, W. and Miller, L. (1967) ***Theory of Scheduling***. Addison-Wesley.

Cook, W., Espinoza, D. G., Goycoolea, Marcos (2007) ***Computing with Domino-Parity Inequalities for the TSP***. INFORMS J. on Computing, vol. 19, No. 3, Summer 2007, pp. 356-365.

Corne, David W. Knowles, Joshua D. and Oates, Martin J. (2000). ***The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization.*** In Marc Schoenauer, Kalyanmoy Deb, G□unter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, editors, Proceedings of the Parallel Problem Solving from Nature VI Conference, pages 839-848, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

Corne, David W. Jerram, Nick R. Knowles, Joshua D. and Oates, Martin J. (2001). ***PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization***. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), pages 283-290, San Francisco, California, 2001. Morgan Kaufmann Publishers.

Czyzak, P. and Jaszkiewicz, A. (1997). ***The multiobjective metaheuristic approach for optimization of complex manufacturing systems*****.** In G. Fandel and T. Gal, editors, Multiple Criteria Decision Making. Proceedings of the XIIth International Conference*,* pages 591-592, Hagen, Germany. Springer-Verlag.

Dantzig G., Fulkerson R., and Johnson S. (1954), **Solution of a large-scale traveling-salesman problem**, Operations Research 2, 393-410.

Das I. and Dennis J. (1998), ***Normal-boundary interaction: A new method for generating the pareto surface in nonlinear multicriteria optimization problems***, SIAM Journal of Optimization, vol. 8, pp. 631-657.

Deb, K., Agrawal, S., Pratab, A. e Meyarivan, T. (2000) ***A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II***. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, editors, Proceedings of the Parallel Problem Solving from Nature VI Conference, pages 849-858, Paris, France. Springer. Lecture Notes in Computer Science No. 1917.

Deb, K. (2001) ***Multi-objective optimization using evolutionary algorithms***, John Wiley & Sons, New York.

Delorme, X., Gandibleux, X. and Rodriguez J. (2003) ***Résolution d'um problème d'évalution de capacite d'infrastructure ferroviaire***. Actes du colloque surl'innovation technologique pour lês transports terrestres (TILT) 2, 647-654. GRRT, Lille, France.

Ehrgott, M. (2000) ***Approximation algorithms for combinatorial multicriteria optimization problems,*** International Transactions in Operational Research, vol. 7, pp. 5-31.

Ehrgott M. & Gandibleux X. (2000) ***A survey and annotated bibliography of multicriteria combinatorial optimization***, OR Spektrum 22: 425-460.

Ehrgot, M. and Ryan, D. M. (2002) ***Constructing robust crew schedules with bicriteria optimization***. Journal of Multi-Criteria Decision Analysis 11, 139-150.

Erickson, Mark, Mayer, Alex, and Horn, Jeffrey (2001). ***The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems.*** In EMO, pages 681-695.

Ferreira, P. A.V. (1999) ***Otimização multiobjetivo: teoria e aplicações***, Universidade Estadual de Campinas, Campinas-SP.

Fischer, R., Richter, k. (1982). ***Solving a Multiobjective Traveling Salesman Problem by Dynamic Programming***. Mathematische Operationsforschung um Statistik, Series Optimization 13(2):247-252.

Fonseca C. M., Fleming P. J. (1993) ***Genetic algorithms for multiobjective optimization: formulation, discussion and generalization***, Procedings of the Fifth Internacional Conference on Genetic Algorithms, San Mateo, Morgan Kauffman Publishers, Califórnia, EUA.

Fresleben, B., Merz, P. (1998). ***A genetic local search algorithm for travelling salesman problem***. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (Eds._, Proceedings of the 4th Conference on Parallel Problem Solving from Nature - PPSN IV. pp.890-900

Glover, F. (1990), ***Tabu Search a tutorial***. USA: Center of Applied Artificial intelligence-University of Colorado.

Godart, J. M. (2001). ***Problèmes d´optimisation combinatoire à caractère économique dans le secteur du tourisme (organization de voyages)***. Ph.D. dissertation, Université de Mons-Hainaut.

Goemans, M. and Kleinberg, J. (1998). ***An improved approximation ratio for the minimum latency problem***. Matemathical Programming, 82:114-124.

Goldbarg, M. C. e Luna, H. P. L. (2000) ***Otimização combinatória e programação linear modelos e algoritmos***. Editora Campus, Rio de Janeiro – RJ.

Goldberg D. E., Richardson J. (1987) ***Genetic algortithm with sharing for multimodal function optimization***. In J. J. GREFENSTETTE Ed., Genetic Algorithm and their aplications: Proceeding of Second Internacional Conference on Genetic Algorithms.

Goldberg, D. E. (1989) ***Genetic algorithms in search, optimization, and machine learning***, Addison-Wesley Publishing Company, Inc., Reading, MA.

Gupta, A., Warburton, A. (1986) ***Approximation Methods for Multiple Criteria Traveling Salesman Problems***. In: Sawaragi, Y (ed.): Towards Interactive and Intelligent Decision Suport Systems: Proccedings of the 7th International Conference on Multiple Criteria Decision Making. Springer-Verlag Berlin 211-217.

Haimes, Y., Lasdon, L. e Wismer, D. (1971), ***On a bicriterion formulation of the problems of integrated system identification and system optimization***, IEEE Transactions on Systems, Man, and Cybernetics 1(3), 296–297.

Hajela, P. and Lin, C. Y. (1992) ***Genetic search strategies in multi-criterion optimal design.*** Structural Optimization, 4:99-107.

Hansen, Michael Pilegaard (1997). ***Tabu search in multiobjective optimisation: MOTS*****.** In *Proceedings of MCDM'97*, Cape Town, South Africa, January.

Hansen, M. P. (2000) ***Use of substitute scalarizing Functions to Guide a Local Search Based Heuristics: The Case of MOTSP***. Journal of Heuristics 6 419-431.

Horn, J., Nafpliotis, N. e Goldberg, D. E. (1994), ***A niched pareto genetic algorithm for multiobjective optimization***, in 'Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence', Vol. 1, IEEE Service Center, Piscataway, New Jersey, pp. 82–87.

Hongyun, Meng and Sanyang, Liu (2003). ***ISPEA: Improvement for the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization with Immunity***. In Proceedings of the Fifth International Conference on

Computational Intelligence and Multimedia Applications (ICCIMA'03), pages 368-372. IEEE Computer Society, September 2003.

Ishibuchi, H. and Murata, T. (1998) ***A multi-objective genetic local search algorithm and its application to flowshop scheduling***, IEEE Transactions on Systems, Man and Cybernetics-part C: Applications and Reviews, 28, 392-403.

Jaszkiewicz, A. (2001) ***Multiple objective methaheuristic algorithms for combinatorial optimization***, Habilitation thesis, 360, Poznan University of Technology, Poznan.

Jaskiewicz, A. (2002) ***Genetic local search for multi-objective combinatorial optimization.*** European Journal of Operational Research, 137, 50-71.

Kennedy, J. and Eberhart R.C. (1995). ***Particle swarm optimization***. In Proc. IEEE Int.'l Conf. on Neural Networks, volume IV, pages 1942–1948, Piscataway, NJ. IEEE Service Center.

Kennedy, J. (1997) ***The particle swarm: social adaptation of knowledge***, In Proceedings of the 1997 Interational Conference on Evoluionary Computation, IEEE, NJ, pp. I303-308.

Kim, Mifa, Hiroyasu, Tomoyuki, Miki, Mitsunori and Watanabe, Shinya (2004). ***SPEA2+: Improving the Performance of the Strength Pareto Evolutionary Algorithm 2***. In Parallel Problem Solving from Nature – PPSN VIII, pages 742-751, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

Kirkpatrick, S., C. D. Gellart Jr., M. P. Vecchi (1983). ***Optimization by simuled annealing***. Science, v. 220, n. 4598, p. 671-680.

Kita, H., Yabumoto, Y., Mori, N. and Nishikawa, Y. (1996). ***Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm***.

Laumanns, M. Rudolph, G. and Schwefel, H. P. (1998). ***A Spatial Predator-Prey Approach to Multi-objective Optimization: A Preliminary Study***. Proceedings of the Parallel problem solving from Nature, V. 241-249.

Lecture Notes In Computer Science; Vol. 1141. Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, p. 504-512

Knowles, Joshua and Corne, David (2000a). ***Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy***. Evolutionary Computation, 8(2):149-172.

Knowles, Joshua and Corne, David (2000b). ***M-PAES: A Memetic Algorithm for Multiobjective Optimization***. In 2000 Congress on Evolutionary Computation, volume 1, pages 325-332, Piscataway, New Jersey, July 2000. IEEE Service Center.

Knowles, Joshua (2004). ***ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems***. Technical Report TR-COMPSYSBIO-2004-01, University of Manchester, September 2004.

Kunkle D. (2005), ***A summary and comparison of MOEA algorithms***. Northeastern University (NU) in Boston (raport), Massachusetts.

Melamed, I. I., Sigal, I. K. (1997) ***The Linear Convolution of Criteria in the Bicriteria Traveling Salesman Problem***. Computational Mathematics and Mathematical Physics 37:8 902-905

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. n., Teller, A. H. e Teller, E. (1953). ***Equation of State Calculations by Fast Computing Machine***, Journal Chemical and Physycal, 21, 1087-1092.

Moore, J. and Chapman, R. (1999) ***Application of particle swarm to multiobjective optimization***, Department of Computer Science and Software Engineering, Auburn University, (Unpublished manuscript).

Nam D. Park C. H. (2000) ***Multiobjective simulated annealing: a comparative study to evolutionary algorithms***, International Journal of Fuzzy Systems, Vol. 2, No. 2, pp. 87-97.

Paquete, L., Stützle, T. (2003) ***A Two-Phase Local Search for the Biobjective Traveling Salesman Problem***. Lecture Notes in Computer Science, Vol 2632, 69, Springer.

Paquete, L. Chiarandini, M. and Stützle, T. (2004) ***Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study***. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, Metaheuristics for Multiobjective Optimisation, volume 535 of Lecture Notes in Economics and Mathematical Systems. Springer Verlag, 2004. ((c) Springer Verlag).

Pareto, V. (1896) ***Cours d'economie politique***, Volume I eII. F.Rouge, Lausanne.

Pulido, Gregorio Toscano and Coello, Carlos A. Coello (2003). ***The micro genetic algorithm 2: Towards online adaptation in evolutionary multiobjective optimization***. In EMO, pages 252-266.

Rudolph, G. (1998). ***Evolutionary Search for Minimal Elements in Partially Ordered Finite sets***. Evolutionary programming VII. 345-353.

Schaffer, J. D. (1984) ***Multiple objective optimization with vector evaluated genetic algorithms***. Ph.D. dissertation, Vanderbilt University.

Sigal, I. K. (1994) ***Algorithm for Solving the Two-Criterion Large-scale Traveling Salesman Problem***. Computational Mathematics and Mathematical Physics 34:1 33-43

Steuer R. E. and Choo R.-U. (1983), ***An interactive weighted tchebycheff procedure of multiple onjective programming***, Mathematical Programming, vol. 26, pp. 326-344.

Steuer R. (1986), ***Multiple criteria optimization: theory, computation and application***. New York, John Wiley & Sons, Inc.

Srinivas, N. e Deb, K. (1994) ***Multiobjective optimization using nondominated sorting in genetic algorithms***, Evolutionary Computation 2(3), 221–248.

Ticona, W. C. G. (2003) ***Aplicação de algoritmos genéticos multi-objetivo para alinhamento de seqüências biológicas***, Dissertação, Universidade de São Paulo, São Carlos-SP.

Thiongane, A. Nagih and Plateau, G. (2001) ***Marie: Metaheuristics associated with reoptimization for integer programming evaluating instances***. Research Report LIPN.

Tung, C. T. (1994) ***A Multicriteria Pareto-optimal Algorithm for the Traveling Salesman Problem***. Ásia-Pacific Journal of Operational Research 11 103-115

Ulungu, E.L., Teghem, J. and Fortemps, Ph. (1995). ***Heuristics for multi-objective combinatorial optimization by simulated annealing***. In J. Gu, G. Chen, Q. Wei, and S. Wang, editors, *Multiple Criteria Decision Making:* Theory and Applications. Proceedings of the 6th National Conference on Multiple Criteria Decision Making, pages 228-238, Windsor, UK. Sci-Tech.

Veldhuizen, V. D. A. and Lamont, G. B. (2000). ***Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art***. Evolutionary Computation, 8(2), 125-147.

Zadeh, L. A. (1994) ***Fuzzy Logic, Neural Networks and Soft Computing***, Commumincations Of ACM, vol. 37.

Zitzler E.,Thiele L. (1998) ***An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach***, Technical Report 43, Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology, Zurich, Suíça.

Zitzler E., Laumanns M., Thiele L. (2001) ***SPEA2: Improving the Strength Pareto Evolutionary Algorithm***, Technical Report 103, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology Zurich, Zurich, Suíça.

***Traveling Salesman Problem***. http://www.math.princeton.edu/tsp/
Ultimo acesso em 10 de fevereiro de 2008

***Traveling Salesman Problem.***
http://users.encs.concordia.ca/~chvatal/tsp/tsp.html
Ultimo acesso em 11 de março de 2008