# Trabalho de Conclusão de Curso

## Otimização Multi-Objetiva Por Enxame de Partículas

Tiago Moreira Trocoli da Cunha

Orientador: Prof. Alexandre Cláudio Botazzo Delbem

**Resumo:** Este projeto teve como objetivo estudar otimização multi-objetiva, pesquisar técnicas de algoritmos evolutivos e inteligência de enxame no seu estado da arte para desenvolver uma meta-heurística baseada em enxame de partículas.

**Palavras-Chave:** Otimização multi-objetiva, MOPSO, particle swarm optimization, algoritmos evolutivos. inteligência de enxame.

**Abstract:** The intention of this project was to study multi-objective optimization, research state-of-the-art evolutionary algorithms techniques and swarm intelligence to develop a multi-objective metaheuristic based on particle swarm optimization.

**Keywords:** Multi-objective optimization, MOPSO, particle swarm optimization, evolutionary algorithm, swarm intelligence.

**The content of this report are sole responsibility of the authors.**
**O conteúdo deste relatório é de única responsabilidade dos autores.**

Departamento de Sistemas de Computação
Instituto de Ciências Matemáticas e de Computação | Universidade de São Paulo |
Av. Trabalhador São-carlense, 400 · Centro · São Carlos/SP · CEP 13566-590 · Brasil ·
www.icmc.usp.br

**Sumário**

# Capítulo 1: Introdução

## 1.1 Contextualização e Motivação

Otimização se refere a encontrar a melhor solução de um problema dado um conjunto de limitações [4]. Começou a ser muito utilizado em pesquisa operacional nas forças armadas em alguns países durante a Segunda Guerra Mundial [5], e depois foi sendo introduzida em diversas áreas, entre elas, na engenharia, na indústria, no setor empresarial, setor financeiro, jogos, robótica, inteligência artificial, controle e economia.

Ao lidar com problemas do mundo real, muitos são intratáveis, ou seja não existe um algoritmo que resolve o problema em tempo polinomial, que encontre uma solução ótima, alguns podem ser bem complexos para modelar, outros podem apresentar um nível de complexidade tal que inviabilize seu uso em certos tipos de aplicações, como de tempo real. Para tais problemas, são usados heurísticas, que são métodos para buscar soluções "aproximadas" do ótimo global, mas sem ter garantia que encontrará uma "boa solução". Ainda existem as meta-heurísticas, que são heurísticas genéricas que resolvem um conjunto de problemas de otimização [6]. Dentro deste contexto, as meta-heurísticas bio-inspiradas vem rapidamente ganhando aceitação dos pesquisadores de uma grande variedade de disciplinas, devido a sua robustez e poder exploratório, e estão sendo aplicadas em problemas em diversas áreas, incluindo multi-objetivo [3].

Elas são bio-inspiradas, porque são inspiradas na biologia, seja na evolução darwiniana, inteligência de coletiva, sistema imunológico e etc. Existem uma variedade de meta-heuristicas bioinspiradas, entre elas, é comumente encontrado na literatura o algoritmo genético(GA), otimização por enxame de partículas *(PSO)*, Otimização por colônia de formigas, programação genética, estratégia evolutiva e programação evolutiva.

Vale ressaltar que a pesquisa de algoritmos bio-inspirados multi-objetivo é ainda recente, o uso do conceito de pareto é ainda mais recente, por exemplo, o NSGA II foi desenvolvido em 2002 por K. Deb et al.[10], o MOPSO, foi pensado por Carlos Coelho e Lechuga também em 2002 [12], o SPEA2 desenvolvido por Zitzler, Laumanns e Thiele em 2001 [21]. Isto serve de motivação para pesquisa nesta área, pois ainda é recente, mas de uma importância imensurável dado que, Otimização multi-objetiva é aplicada em uma grande variedade de áreas [19].

## 1.2 Objetivos

O primeiro objetivo do trabalho de conclusão de curso foi estudar otimização multi-objetiva juntamente com os algoritmos no estado da arte e suas técnicas para lidar com problemas multimodais, de convergência e diversidade das solução. O outro, foi desenvolver um algoritmo baseado em inteligência de enxame para lidar com otimização multi-objetiva.

### 1.3 Organização do Trabalho

No que tange a organização do trabalho, o tópico de revisão bibliográfica foi deixado no capítulo dois, para que o leitor possa entender resumidamente o que é inteligência de enxame, os conceitos de otimização multi-objetiva e as técnicas usadas nos algoritmos para lidar com tal problema. Depois do leitor ter conhecimento dessas técnicas e idéias, irá ser apresentado os algoritmos multi-objetivos e por fim os resultados obtidos juntamente com gráficos e tabelas. As considerações finais, obviamente, foram deixadas no final deste trabalho que resume as aprendizagens do autor e trabalhos futuros. Por fim, as referências são apresentadas.

## Capítulo 2: Revisão Bibliográfica

Essa seção apresenta uma breve revisão bibliográfica sobre conceitos de inteligência de enxame, sistema auto-organizável, otimização por enxame de partículas, otimização multi-objetiva, as técnicas distância de aglomeração, arquivo de pareto e ordenação de pareto nesta ordem.

### 2.1 Inteligência de Enxame

Inteligência de Enxame ou SI, do inglês *Swarm Intelligence* originou de estudos de colônias ou de enxames de organismos sociais [16]. Estudos de comportamentos socias de organismos de exames incentivou o desenvolvimento de algoritmos de otimização, de clusterização e aprendizado de máquina.

Por exemplo, a simulação da coreografia de revoadas de pássaros resultou no desenvolvimento do algoritmo de otimização por enxame de partículas(PSO), por Kennedy, Eberhart e Shi [29]. E estudos de comportamentos coletivo de formigas resultou na otimização por colônia de formigas, desenvolvido por Marco Dorigo, na sua tese de Phd na Politecnico di Milano [28] . Sua meta-heurística conseguiu resolver uma série de problemas difíceis em diversas áreas[30] e seus estudos de inteligência de enxame lhe rendeu vários prêmios por suas contribuições em otimização e inteligência artificial [31].

Pode-se formalizar um exame como um grupo de agentes inteligentes que se comunicam entre si, direta ou indiretamente, através de suas ações no ambiente [15]. As interações entre agentes resultam em uma estratégia coletiva de resolução de problemas. Então, SI se refere a comportamentos de resolução de problemas que emergem das interações de agentes, e inteligência de exame computacional (CSI) se refere aos algoritmos que simulam tais comportamentos. Mais formalmente, SI é um propriedade de um sistema através dos quais comportamentos coletivos de agentes que interagem localmente entre si e o ambiente por regras simples, resulta no surgimento de um padrão global funcional e coerente [14]. O objetivo da inteligêcia de exame computacional é simular comportamentos simples de indivíduos e suas interações, com o intuito de se obter um comportamento mais complexo que pode ser usado em resoluções de problemas, principalmente em otimização.

### 2.2 Sistema Auto-Organizável

Um sistema auto-organizável, é uma organização descentralizada, na qual algum forma global de ordem surge através de interações locais entre componentes de um sistema inicialmente desordenado. Este processo pode ser espontâneo, e não é necessariamente

controlado por nenhum agente auxiliar fora do sistema. É frequentemente ativado por ações randômicas e amplificado por todo o sistema. Como é descentralizado, a organização é tipicamente robusta e apta para sobreviver e se recuperar de danos ou perturbações que veem fora dela [43].

Um sistema auto-organizável ocorre em uma variedade de fenômenos físicos e químicos, na biologia e entre outras áreas. Na química, temos a automontagem molecular, redes auto-catalíticas e sistemas de reações-difusão. Na física, temos a magnetização espontânea, a cristalização, supercondutividade, condensação de Bose-Einstein e muitos outros fenômenos . Na biologia, temos redes neurais, organização de agentes sociais como enxame de insetos e tantos outros [45].

Todo sistema auto-organizável apresenta estas três características [44]:

1) O surgimento de uma ordem global através de múltiplas interações entre seus componentes.
2) Controle distribuído.
3) O sistema é não linearmente dinâmico em relação a alguma resposta(“feedback”) positiva ou negativa.

## 2.3 Otimização por Enxame de Partículas

O PSO ,do inglês *Particle Swarm Optimization*, é um processo de busca baseada em população, onde indivíduos(ou agentes) são chamados de partículas e sua população de enxame. Existem dois tipos clássicos de PSO, o local e global.

No Local-PSO, cada partícula “voa” pelo espaço de busca multidimensional, ajustando sua posição no espaço de acordo com sua própria experiência e também das partículas vizinhas. Então, uma partícula, pode fazer uso da melhor posição encontrada por ela e pela sua vizinhança e assim ajustar sua posição em direção a solução ótima. O efeito disso é que, as partículas “voam” em direção ao ótimo, enquanto rondam uma certa área em volta da melhor solução atual.

No ponto de vista da inteligência artificial, as partículas são agentes inteligentes. Elas aprendem a “voar” no espaço de acordo com suas próprias experiências, então elas são dotadas de memória, além disso, se “comunicam” entre si e “imitam” as outras, neste caso daquela ou aquelas que estão “voando melhor”, ou seja, elas são agente sociais. Daí porque a meta-heurística se enquadra na inteligência de enxame ou coletiva, a ideia intuitiva aqui é que a imersão ou aprofundamento da aprendizagem coletiva faz emergir um padrão global ótimo.

No ponto de vista matemático, a melhor experiência(ou melhor solução) encontrada pela partícula seria um limitante inferior, enquanto a “imitação” da melhor partícula seria o limitante superior. Cada partícula ronda o limitante superior e inferior, e a proporção que vão aprendendo a “voar” no espaço, ou seja a otimização vai convergindo, estes dois limitantes vão ficando cada vez mais próximos até se encontrarem. A prova de sua convergência é descrito em vários artigos,

entre eles do pesquisador Dong ping Tian [32].

Resumidamente, cada partícula é tratada como um ponto de um espaço com m dimensões. A partícula i na iteração t, é representada como $x_i(t) = (x_{i1}, ..., x_{im})$. A melhor posição individual encontrada pela partícula i até a iteração t, é recordada e representada por $y_i(t) = (y_{i1}, ..., y_{im})$, esta componente faz parte da experiência dela, é o componente cognitivo. A melhor posição encontrada da vizinhança da partícula i até a iteração t é $\hat{y}_i(t) = (\hat{y}_{i1}, ..., \hat{y}_{im})$,esta componente representa a comunicação entre elas, ou seja, é o componente social. E a velocidade dela i é representada por $v_i(t) = (v_{i1}, ..., v_{im})$. A partícula é manipulada de acordo com a equação abaixo:

$$v_{ij}(t+1) = w.v_{ij}(t) + c_1.r_{1j}.(y_{ij}(t) - x_{ij}(t)) + c_2.r_{2j}.(\hat{y}_{ij}(t) - x_{ij}(t)) \quad (2.1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2.2)$$

Onde j( $1 \leq j \leq m$ ) é a dimensão, $c_1$ e $c_2$ são constantes positivas chamadas de aceleração, usadas para escalar componentes cognitivos e sociais, respectivamente. E $r_{1j}$, $r_{2j} \sim U(0,1)$ são variáveis aleatórias entre 0 e 1 de uma distribuição uniforme. Estas variáveis servem para introduzir elementos estocásticos no algoritmo.

A vizinhança de cada partícula é representada por um grafo, em que os vértices são as partículas e as arestas a vizinhança . Se considerarmos o problema de minimização, a melhor posição da vizinhança da partícula até a iteração t+1 é:

$$\hat{\mathbf{y}}_i(t+1) \in \{\mathcal{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \min\{f(\mathbf{x})\}, \quad \forall \mathbf{x} \in \mathcal{N}_i\}$$

Em que $N_i$ é vizinhança da partícula, e é definida como:

$$\mathcal{N}_i = \{\mathbf{y}_{i-n_{\mathcal{N}_i}}(t), \mathbf{y}_{i-n_{\mathcal{N}_i}+1}(t), \ldots, \mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+n_{\mathcal{N}_i}}(t)\}$$

A melhor posição individual da partícula até a iteração t+1, é definida assim:

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases}$$

O algoritmo é descrito a seguir:

Local - PSO

Crie e inicialize uma população com n agentes de m dimensões
**repita**
*// Cada agente determina sua melhor posição individual*

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases}$$

*// Cada agente determina a melhor posição da sua vizinhança*

$$\hat{\mathbf{y}}_i(t+1) \in \{\mathcal{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \min\{f(\mathbf{x})\}, \quad \forall \mathbf{x} \in \mathcal{N}_i\}$$

*//Cada agente se desloca no espaço.*
Aplique a equação (2.1) e (2.2)
**até** *a condição de parada for verdadeira*;

No caso do Global-PSO, não existe vizinhança. Então em vez de cada partícula memorizar a melhor solução na sua vizinhança até a iteração t+1, o enxame memoriza a melhor solução(global) encontrada até a iteração t+1, logo:

$$\hat{\mathbf{y}}(t) \in \{\mathbf{y}_0(t), \ldots, \mathbf{y}_{n_s}(t)\} | f(\hat{\mathbf{y}}(t)) = \min\{f(\mathbf{y}_0(t)), \ldots, f(\mathbf{y}_{n_s}(t))\}$$

Global - PSO

Crie e inicialize uma população com n agentes de m dimensões
**repita**
*// Cada agente determina sua melhor posição individual*

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases}$$

*// O enxame determina sua melhor posição global*

$$\hat{\mathbf{y}}(t) \in \{\mathbf{y}_0(t), \ldots, \mathbf{y}_{n_s}(t)\} | f(\hat{\mathbf{y}}(t)) = \min\{f(\mathbf{y}_0(t)), \ldots, f(\mathbf{y}_{n_s}(t))\}$$

*//Cada agente se desloca no espaço.*
Aplique a equação (2.1) e (2.2)
**até** *a condição de parada for verdadeira*;

## 2.4 Otimização Multi-Objetiva

Otimização multi-objetiva, também chamada de Otimização multi-critéria pode ser

escrita como se segue [18]:

$$f : \Re^m \rightarrow \Re^n, \quad min\, f(x) = min\, [f1(x), f2(x), \ldots, fn(x)]^T$$
$$sujeito\ a:$$
$$G_i(x) = 0,\ i = 1, ..., k$$
$$H_i(x) \leq 0,\ i = 1, ..., l$$

Sendo que, se alguma função é maximizada, é equivalente a minimizar sua forma negativa, ou seja, max [fk(x)] = min [-fk(x)]. O $n \in N$ e $n \geq 2$ é o número de funções objetivas e o X é o conjunto viável, o elemento x* $\in$ X é chamado de solução viável, o vetor $z^* := f(x^*) \in R$ é chamado de vector objetivo ou vetor resultado.

Em otimização multi-objetiva, geralmente não existe uma solução viável que minimiza todas as funções objetivo. Então, foi criado o conceito de Pareto, dado a seguir:

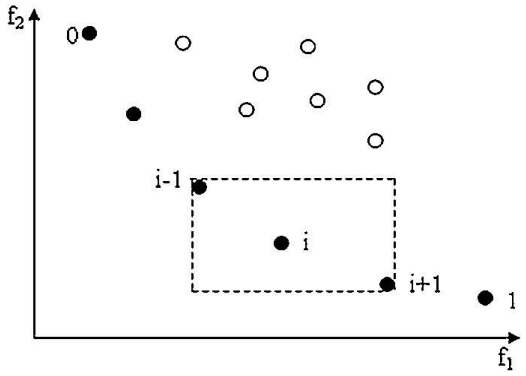
- Dominação: Uma solução viável x* domina uma outra x se as seguintes condições forem satisfeitas: $f_i(x^*) \leq f_i(x), \forall\ i \in \{1, ..., n\}$ e $\exists\ j \in \{1, ..., n\}$ tal que $f_j(x^*) < f_j(x)$. Neste caso, a notação matemática é $x^* > x$.
- Dominação fraca: Uma solução viável x* domina fracamente uma outra x se $f_i(x^*) \leq f_i(x), \forall\ i \in \{1, ..., n\}$ . Cuja a notação é $x^* \succcurlyeq x$.
- Não dominado: Uma solução x* se diz não dominada se não existe uma solução viável x, tal que, ela domina x*.
- Conjunto Ótimo de Pareto: É o conjunto de todo x* que não é dominado. Seja $P_s$ tal conjunto, então:
  $P_s = \{x^* \in X : [\neg \exists\ x\ : (f_i(x^*) \leq f_i(x), \forall\ i \in \{1, ..., n\}\ e\ \exists\ j \in \{1, ..., n\}\ f_j(x^*) < f_j(x))]\}$
- *Pareto Front*: É o conjunto de todos os vetores f(x) correspondente ao conjunto ótimo de pareto. Seja, $P_f$ o *pareto front*, então: $P_f = \{f(x^*)\ :\ x^* \in P_s\}$.

No caso de Otimização com uma função, só existe um objetivo, que é encontrar a solução ótima global. Mesmo quando tal Otimização é multimodal muitos algoritmos só procuram uma solução ótima [20]. No caso de Otimização multi-objetiva, existem várias soluções ótimas, então o objetivo de procurar uma única solução não é adequado. O objetivo se torna dois, de maximizar o número de soluções não dominadas(convergência) e maximizar a diversidade.

## 2.5 Distância de Aglomeração

A distância de aglomeração é uma medida que calcula a densidade de um ponto no espaço em relação a seu conjunto de pontos. O cálculo é feito ao se calcular o semiperímetro do

cubóide do ponto i. Primeiro ordena-se de forma ascendente os pontos em relação a cada função, diremos que existem M funções. O cubóide é a região formada pelos vértices dos pontos imediatamente antecessor e predecessor de i em relação a ordem de pontos em cada função. Assim com M funções o cubóide terá 2M vértices. A figura a seguir ilustra isto:




(2.4) - Distancia de Aglomeração

Seja U=$\{p_1, ..., p_n\}$ o conjunto de pontos

Seja $d_i = 0 \ \forall i \in \{1, 2 .., N\}$ a distância de aglomeração

*para cada função* m $\in$ M *faça*:

U = ordernar(U,m) // *Ou seja,* $f(p_1)_m \leq f(p_2)_m ... \leq f(p_n)_m$

$d_1 = d_1 + 1.0$

$d_n = d_n + 1.0$

*para* i = 2 *até* n-1 *faça*:

$$d_i = d_i + (f(p_{i+1})_m - f(p_{i-1})_m)/(f(p_n)_m - f(p_1)_m)$$

Os pontos extremos tem densidade infinita, mas nos algoritmos do autor eles tem densidade um(1.0), e obviamente os não extremos tem densidade menor que um. Vale ressaltar, que na figura os pontos i-1 e i+1 são próximos de i em relação a $f_2$.

## 2.6 Arquivo de Pareto

O arquivo de pareto é um conjunto de pontos não dominados. Ou seja, defini-se $P = \{p_1, ..., p_n\}$ o conjunto de pontos e Q os pontos não dominados, para cada elemento de $P$, digamos $p_i$, $p_i \in Q$, se $\neg \exists \ p_j$ tal que $p_i < p_j$. O algoritmo G-MOPSO e L-MOSPSO utiliza o arquivo, então para cada iteração destes algoritmos a atualização do arquivo é feita comparando os elementos do arquivo com os novos pontos encontrados pelos agentes. O algoritmo ingênuo para atualização é simples. Seja $Q$ um conjunto não dominado na iteração até a $t$ e $P$ o

conjunto de pontos encontrados na iteração $t+1$, cada ponto de $Q = Q \cup P$ é comparado com todos os outros, caso ele for dominado por algum ponto, ele é removido de $Q$ (arquivo).

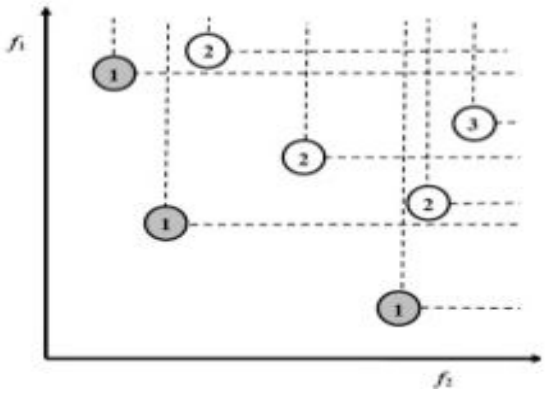
O autor usou outro método descrito a seguir [34] :

$P = \{p_1, ..., p_n\}$, os pontos encontrados na iteração $t+1$
$Q = \{q_1, ..., q_n\}$, os pontos não dominados na até a iteração $t$.

(2.5) - Atualização:
**Passo 1:** Faça $P = Q \cup P$ e inicialize seu contador, $i = 2$. Faça $Q = \{q_1\}, q_1 = p_1 \in P$ .
**Passo 2:** Inicialize o contador de $Q$, $j = 1$.
**Passo 3:** Compare os pontos $q_j$ e $p_i$.
**Passo 4:** Se $q_j < p_i$ :
Faça $Q - \{q_j\}$.
Se $j < |Q|$, faça $j = j + 1$ e vá ao passo 3.
Se não, vá ao passo 5.
Se $p_i < q_j$ :
Faça $i = i + 1$ e vá ao passo 2.
**Passo 5:** Faça $Q = Q \cup \{p_i\}$. Se $i < |P|$, faça $i = i + 1$ e vá ao passo 2. Se não, pare e declare $Q$ como conjunto não dominado.

## 2.7 Ordenação de Pareto

O arquivo de pareto só contém as soluções não dominadas. No entanto, o algoritmo G-MOPSO2 utiliza a ordenação de pareto em que ordena as soluções em níveis em relação a dominância. Então, o primeiro nível contém as soluções não dominadas. O segundo nível contém as soluções não dominadas tirando as soluções do primeiro nível. O terceiro nível, igualmente só que tirando as soluções do primeiro e segunda nível. E assim sucessivamente.



A imagem acima mostra um exemplo da ordenação de pareto para o problema de minimização. O algoritmo é descrito a seguir [35]:

Seja $P$ o conjunto de pontos.
Seja $F_i$ o conjunto de pontos de nível $i$.

(2.6) - Algoritmo Ordenação de Pareto (2.6):

Para cada $p \in P$ :
    $S_p = \{\emptyset\}$
    $n_p = 0$
    Para cada $q \in P$ : *Se p domina q*
        Se $q \prec p$ então:
            $S_p = S_p \cup \{q\}$ *Adicione q no conjunto dominado por* p
        Mas se $p \prec q$, então:
            $n_p = n_p + 1$ Incremente o contador de dominância de p
    Se $n_p = 0$ então: *p pertence ao primeiro pareto front*
        $p_{rank} = 1$
        $F_1 = F_1 \cup \{p\}$
$i = 1$ *Inicialize o contador do pareto front*
Enquanto $F_i \neq \emptyset$ :
    $Q = \{\emptyset\}$ *Usado para armazenar os pontos do próximo front*
    Para cada $p \in F_i$ :
        Para cada $q \in S_p$ :
            $n_q = n_q - 1$
        Se $n_q = 0$ então: *q pertence ao próximo front*
        $q_{rank} = 1$
        $Q = Q \cup \{q\}$
    $i = i + 1$
    $F_i = Q$

## Capítulo 3: Desenvolvimento do Trabalho

### 3.1 Projeto

Como dito anteriormente o objetivo do projeto foi desenvolver uma meta-heurística bioinspirada multi-objetiva. Tal meta-heurística escolhida foi baseada no PSO. Foram desenvolvidos três MOPSO(*Multiobjective Particle Swarm Optimization*), que foi dado o nome de G-MOPSO, L-MOPSO e G-MOPSO2.

Para cada algoritmo, foram realizados 100 testes de trezentas iterações de cada um dos já conhecidos *benchmarks* na literatura, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 e FON , com a finalidade de comparar a convergência e a diversidade das soluções dos mesmo. Os *benchmarks* ZDT foram desenvolvidos no Departamento de Engenharia Elétrica do Instituto Tecnológico de Zurique, Suíça, pelos pesquisadores E. Zitzler, L. Thiele e D. Deb [36] e posteriormente testes mais novos, os DTLZ [39]. O autor deste projeto observou que são comumente usados nos artigos científicos para testar as meta-heurísticas. Segundo estudos [37], as meta-heurísticas no estado da arte nos anos 2000 não conseguiram convergir para o conjunto de solução global ótimo, exceto pelo SPEA, e nenhuma conseguiu este feito no ZDT4 evidenciando assim a dificuldade que estes *benchmarks* oferecem aos algoritmos. Posteriormente o SPEA2 e NSGA2, que são aprimoramentos dos algoritmos já criados, convergiram nas soluções globais ótimas. O FON foi desenvolvido pelos pesquisadores Fonseca e Fleming[40].

O ZDT1, ZDT2, ZDT3 e ZDT4 tiveram trinta variáveis, ZDT6 foram dez, e FON por definição tem três. As funções estão descritas a seguir:

| Nome | *Benchmarks* | Domínio | Tipo do Problema |
|---|---|---|---|
| ZDT1 | $f_1(x) = x_1$<br>$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}}]$<br>$g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ | $x_i \in [0,1]$<br>$i = 1...n$ | convexo |
| ZDT2 | $f_1(x) = x_1$<br>$f_2(x) = g(x)[1 - (\frac{x_1}{g(x)})^2]$<br>$g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ | $x_i \in [0,1]$<br>$i = 1...n$ | não convexo |
| ZDT3 | $f_1(x) = x_1$<br>$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)} sin(10\pi x_1)]$ | $x_i \in [0,1]$<br>$i = 1...n$ | convexo e descontínuo |

| | | | |
|---|---|---|---|
| | $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ | | |
| ZDT4 | $f_1(x) = x_1$<br>$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}}]$<br>$g(x) + 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10cos(4\pi x_i))$ | $x_1 \in [0,1]$<br>$x_i \in [-5,5]$<br>$i = 1...n$ | não convexo |
| ZDT6 | $f_1(x) = 1 - exp(-4x_1)[sin(6\pi x_1)]^6$<br>$f_2(x) = g(x)[1 - (\frac{x_1}{g(x)})^2]$<br>$g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)^{0.25}$ | $x_i \in [0,1]$<br>$i = 1...n$ | não convexo, não uniforme e espaçado |
| FON | $f_1(x) = 1 - exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2)$<br>$f_2(x) = 1 - exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2)$ | $x_i \in [-4,4]$<br>$i = 1...3$ | Não convexo |

Foram utilizadas três métricas. Uma pertence a métrica de convergência que o autor deste artigo traduziu como taxa de acerto, proposta no artigo [22]. Em que |S| é a quantidade das soluções encontradas e |P| é quantidade das soluções encontradas que pertencem ao pareto front.

$$T = \frac{|P|}{|S|}$$

A outra, é uma métrica de diversidade das soluções. Proposta por Zitzler et al. , é chamada de hipervolume (HV). Primeiro, é definido um ponto de referência, o autor definiu p = (11,11), depois é calculada para cada solução $i \in S$, um hiperplano $v_i$ relação ao ponto de referência. Para mais detalhes, consulte os artigos [23][24].

$$HV(S) = \bigcup_{i=1}^{|S|} v_i$$

Por último, a métrica de espaçamento sugerido por Schott[25]. Mostrada abaixo, esta métrica mede a distância relativa entre duas soluções consecutivas de $S$, onde $d_i = min(\sum_{m=1}^{M} |f_m^i - f_m^k|), \ \forall k \in S \ e \ k \neq i$ , M é o número de objetivos, e $\bar{d}$ é a média dos $d_i$ .Esta métrica calcula o desvio padrão dos $d_i$, então quanto mais uniformemente espaçadas estiverem as soluções menor será o desvio padrão.

$$\sigma = \sqrt{\frac{1}{|S|}\sum_{i=1}^{|S|}(d_i - \overline{d})^2}$$

Independente dos algoritmos, eles preservam as seguintes características:

| | |
|---|---|
| Tamanho da população | : 50 |
| Número de iterações | : 300 |
| Aceleração cognitiva( $c_1$ ) | : 2.0 |
| Aceleração Social( $c_2$ ) | : 2.0 |

### 3.2 Detalhes das Meta-heurísticas

#### 3.2.1 G-MOPSO

O G-MOPSO, foi o primeiro algoritmo criado pelo autor. O "G" vem de Global-PSO, então, basicamente foi implementado usando este algoritmo junto com o arquivo de pareto com a tentativa de manter a convergência e a métrica da distância de aglomeração para manter a diversidade das soluções.

O algoritmo funciona assim, para cada iteração cada agente atualiza sua melhor posição individual usando o conceito de dominância de pareto. As soluções encontradas $x_i(t)$ em cada iteração $t$, são comparadas com as do arquivo de pareto e a partir daí, este é atualizado. A distância de aglomeração é aplicada para cada solução no arquivo de pareto. Por fim, cada agente é guiado por um líder, que é uma das soluções no arquivo de pareto usando a seleção por roleta [26] em relação a sua densidade(calculada usando a distância de aglomeração).

---

G-MOPSO

---

Crie e inicialize uma população com n agentes de m dimensões
Crie e inicialize o arquivo de pareto vazio, $A(t=0)$.
**repita**
*//Cada agente determina a sua melhor posição individual*
$y_i(t+1) = x_i(t),\ se\ y_i(t) \prec x_i(t)$
atualize o arquivo de pareto (2.5)
aplique a distancia de aglomeração no arquivo de pareto (2.4)
*//Cada agente escolhe um líder usando a seleção por roleta.*
$\hat{y}_i(t+1) \sim roulette(\ A(t+1)\ )$
*//Cada agente se desloca no espaço.*
Aplique a equação (2.1) e (2.2)

---

---

**até** *a condição de parada for verdadeira*;

---

### 3.2.2 G-MOPSO2

Este foi a última meta-heurística e foi a que apresentou os melhores resultados, que foram muito superiores as outras duas. O porquê deste ser superior aos outros está apresentado no tópico "Considerações", no final deste capítulo.

O G-MOPSO2, como o nome sugere, é uma meta-heurística baseada no Global-PSO. Ela não utiliza o arquivo de pareto como no seu antecessor, mas todas as melhores soluções individuais e as soluções atuais, $x_i(t)$ e $y_i(t)$ respectivamente, dos agentes na iteração $t$, são submetidas a ordenação de pareto, como descrita no capítulo 2.6. Assim, todos os elementos do conjunto $y(t)$ e $x(t)$ tem um rank. Depois, calcula-se a distância de aglomeração do conjunto $x(t)$ e $y(t)$, então todos os elementos tem uma densidade.

Para cada agente, um líder $\hat{y}_i(t)$ é selecionado aleatoriamente do conjunto $y(t)$ (não inclui $x(t)$) por seleção de torneio [27] em relação ao rank e densidade. Depois, tal líder encontrado pelo agente, é usado como a melhor solução $\hat{y}_i(t)$ para compor a equação da sua velocidade. Além disso, a melhor posição individual do agente i na iteração $t$ é feita por rank e densidade.

---

G-MOPSO2

---

Crie e inicialize uma população com n agentes de m dimensões
**repita**
*//Cada agente determina a sua melhor posição individual*
$y_i(t+1) = x_i(t)$, *se* $(x(t)_{i,rank} < y(t)_{i,rank}) \vee (x(t)_{i,rank} = y(t)_{i,rank} \wedge x(t)_{i,densidade} > y(t)_{i,densidade})$
aplique a ordenação de pareto em $y(t+1)$ (2.6)
aplique a distância de aglomeração em $y(t+1)$ (2.4)
*//Cada agente escolhe um líder usando a seleção por torneio.*
$\hat{y}_i(t+1) \sim tournament(\ y(t+1)\ )$
*//Cada agente se desloca no espaço.*
Aplique a equação (2.1) e (2.2)
**até** *a condição de parada for verdadeira*;

---

### 3.2.3 L-MOPSO

O L-MOPSO foi a segunda meta-heurística criada pelo autor e a que teve o pior resultado dos três. Diferente da anterior, foi baseado no Local-PSO. Posteriormente, é mostrado o porque o

conceito de pareto não funciona bem localmente no caso do PSO. O algoritmo funciona assim, cada agente tem um arquivo de pareto de tamanho dez, este arquivo representa as melhores soluções, $y(t)$, encontradas localmente, ou seja, pela vizinhança. A cada iteração todos os arquivos são atualizados usando o conceito de dominância de pareto. Cada agente é guiada por um líder, $\hat{y}_i(t)$, que pertence a seu arquivo de pareto. Como as outras, é aplicado a distância de aglomeração no arquivo para calcular a densidade das melhores soluções.

L-MOPSO

Crie e inicialize uma população com n agentes de m dimensões
Crie e inicialize o n arquivos de pareto vazio, $A_i(t=0)$.
**repita**
*//Cada agente determina a sua melhor posição individual*
$y_i(t+1) = x_i(t),\ se\ y_i(t) < x_i(t)$
atualize os n arquivos de pareto (2.5)
aplique a distância de aglomeração nos n arquivo de pareto (2.4)
*//Cada agente escolhe um líder do seu arquivo de pareto usando a seleção por roleta.*
$\hat{y}_i(t+1) \sim roulette(\ A(t+1)\ )$
*//Cada agente se desloca no espaço.*
Aplique a equação (2.1) e (2.2)
**até** *a condição de parada for verdadeira*;

### 3.3 Resultados Obtidos

A primeira linha de cada algoritmo representa a média dos cem testes executadas por trezentas iterações, a segunda a variância e a terceira a amplitude.

| Tabela da Taxa de Acerto | | | | | | |
|---|---|---|---|---|---|---|
| Algoritmo | ZDT1 | ZDT2 | ZDT3 | ZDT4 | ZDT6 | FON |
| G-MOPSO | 0.0 | 0.0 | 0.0 | 0.0 | 0.9982 | 0.8742 |
| | 0.0 | 0.0 | 0.0 | 0.0 | 3.276e-05 | 1.536e-3 |
| | 0.0 | 0.0 | 0.0 | 0.0 | 0.02 | 0.18 |
| G-MOPSO2 | 1.0 | 1.0 | 0.9612 | 0.9600 | 0.9882 | 0.8628 |
| | 0.0 | 0.0 | 4.86e-4 | 0.0384 | 1.44e-4 | 2.0967e-4 |

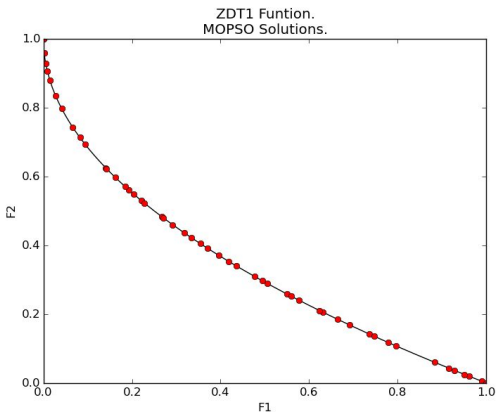
| | | | | | | |
|---|---|---|---|---|---|---|
| | 0.0 | 0.0 | 0.1 | 1.0 | 0.04 | 0.24 |
| L-MOPSO | 0.998 | 1.0 | 0.9454 | 0.0 | 0.941 | 0.0 |
| | 3.6e-05 | 0.0 | 0.00128684 | 0.0 | 0.001579 | 0.0 |
| | 0.02 | 0.0 | 0.02 | 0.0 | 0.02 | 0.0 |

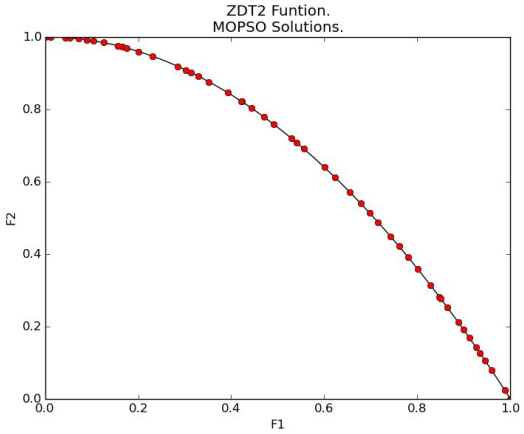
| Tabela de Hipervolume | | | | | | |
|---|---|---|---|---|---|---|
| Algoritmo | ZDT1 | ZDT2 | ZDT3 | ZDT4 | ZDT6 | FON |
| G-MOPSO | 304.579 | 328.141 | 305.734 | 309.31090 | 117.47744 | 119.69910 |
| | 142.1240 | 28.524419 | 146.06684 | 96980.333 | 2.2386e-4 | 0.0275341 |
| | 61,6 | 15,4 | 53,9 | 1432.79 | 0.074 | 0.774 |
| G-MOPSO2 | 120.61637 | 120.145048 | 128.72604 | 118.76314 | 117.50913 | 120.09362 |
| | 0.0686909 | 0.99868853 | 0.0779609 | 16.298670 | 8.926e-07 | 0.1526731 |
| | 1.23 | 5.12 | 2.06 | 10.65 | 0.08 | 2.3285441 |
| L-MOPSO | 120.62187 | 111.881025 | 128.39726 | 122.36130 | 117.44908 | 113.65908 |
| | 8.978e-05 | 14.4018175 | 0.3519352 | 19361.186 | 0.0135800 | 0.1448563 |
| | 0.04 | 10.28 | 3.52 | 912.03 | 0.42 | 2.42 |

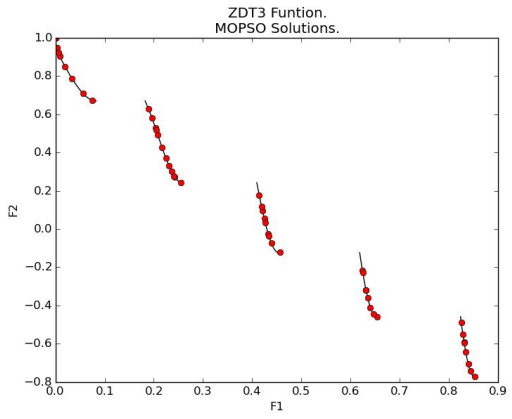
| Tabela de Espaçamento | | | | | | |
|---|---|---|---|---|---|---|
| Algoritmo | ZDT1 | ZDT2 | ZDT3 | ZDT4 | ZDT6 | FON |
| G-MOPSO | 0.0 | 0.0 | 0.0 | 3592183 | 4122.0676 | 66.805594 |
| | 0.0 | 0.0 | 0.0 | 2.225311 | 29183181 | 2851.4203 |
| | 0.0 | 0.0 | 0.0 | 28688729 | 101663.4 | 217.72 |
| G-MOPSO2 | 68.94e-20 | 63.102e-20 | 119.83e-20 | 87.466e-20 | 52.87e-20 | 114.9e-20 |
| | 2436e-40 | 2248.0e-40 | 8582.9e-40 | 38567e-40 | 1312e-40 | 1273e-40 |
| | 269.8e-20 | 233.06e-20 | 357.25e-20 | 1705e-20 | 168.5e-20 | 634.7e-20 |

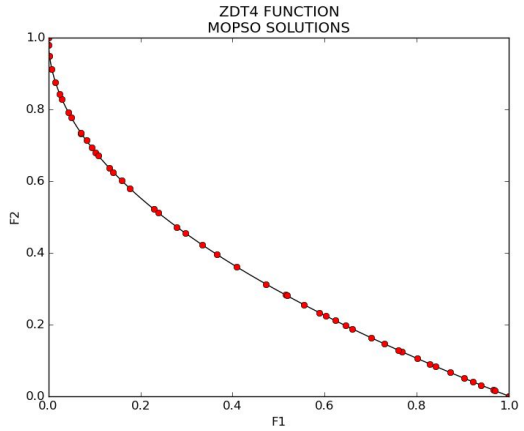
| | | | | | | |
|---|---|---|---|---|---|---|
| L-MOPSO | 95.31e-20 | 287.25e-20 | 1276.8e-20 | 1047071.1 | 12172e-20 | 96.53e-20 |
| | 10650e-40 | 151349e-40 | 11037e-40 | 7.075e+12 | 45127e-40 | 1269e-40 |
| | 748.3e-20 | 8929.9e-20 | 32181e-20 | 18288436 | 13346e-20 | 619.4e-20 |

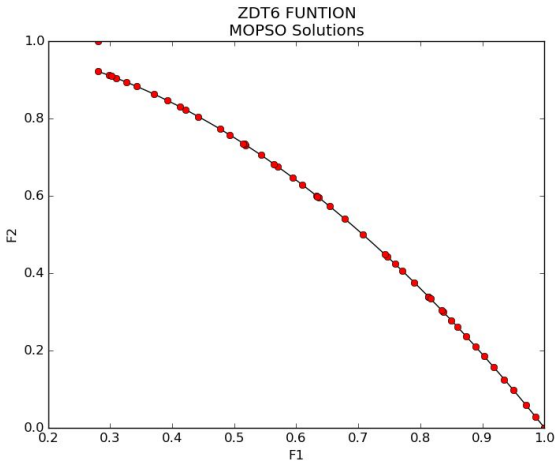
As imagens representam o resultado do G-MOPSO2, os pontos é sua solução encontrada e a curva é o pareto front. Não foram encontrados o pareto front do FON, só o conjunto de pareto.






Como se pode ver nos resultados, o G-MOPSO2 tem uma eficácia "muito superior" aos outros dois. Ele conseguiu uma taxa de acerto média igual a 100% em todos os cem testes do ZDT1 e ZDT2, e valores próximos de 100% no ZDT3, ZDT4 e ZDT6, mas não conseguiu convergir "muito bem" no FON tendo média de 86.28% de acerto. O L-MOPSO conseguiu uma taxa de acerto médio maior que 90% no ZDT1, ZDT2, ZDT3 e ZDT6, apesar disso o hipervolume médio no ZDT2 foi de 111.881025, um valor "muito baixo" significando que as soluções encontradas estão "muito próximas" e de fato, a maior parte das soluções encontradas(pareto front) são (1,0) ou (0,1).

Em relação ao hipervolume, pode-se comparar o G-MOPSO2 com a meta-heurística SPEA2, desenvolvido por pesquisadores do Instituto de Tecnologia de Zurich usando também a mesma quantidade de agentes(50) com 250 iterações [33]. Como se pode ver a seguir, ambos os valores são próximos, isto significa que ambos têm uma dispersão próxima em relação aos *benchmarks* ZDT.

| Hipervolume | | | | | |
|---|---|---|---|---|---|
| | ZDT1 | ZDT2 | ZDT3 | ZDT4 | ZDT6 |
| G-MOPSO2 | 120.6164 | 120.14505 | 128.72604 | 118.76314 | 117.50913 |
| SPEA2 | 120.6574 | 120.32440 | 128.77364 | 120.65745 | 117.51166 |

Mas, porque o G-MOPSO2 teve eficácia superior aos dois? Tanto o G-MOPSO quanto o L-MOPSO usam arquivo de pareto baseado no algoritmo (2.5), o problema dele é que o arquivo só mantem soluções que não foram dominadas, não fazendo uso de mais nenhuma informação. Apesar disso forçar uma "grande" pressão seletiva pode facilmente causar uma convergência prematura. degradando a qualidade da convergência e suprindo a diversidade das soluções.

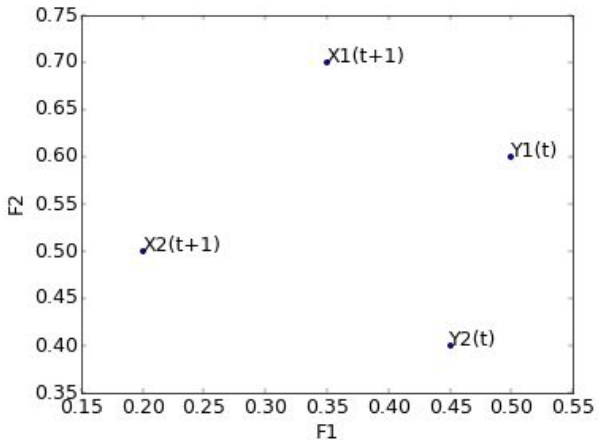
A maneira que o autor lidou com o problema anterior foi usar a ordenação de pareto. Isto usa a informação de dominância e não dominância, além de que as soluções dominadas, com pior rank tem chances de serem líderes. E de fato, isto funcionou. Percebe-se que umas das diferenças entre o G-MOPSO e o G-MOPSO2 é o uso da ordenação de pareto.

O outro ponto, é a seleção por roleta usada também pelo G-MOPSO e L-MOPSO. Como é sabido, esta seleção é tendenciosa em relação as melhores soluções. Ela é imprecisa no caso da distância de aglomeração (2.4), pois a seleção fica tendenciosa nas soluções que estão nos extremos, afinal são estas que têm densidade maior que um(1.0). E de fato, o L-MOPSO convergiu no ZDT2, mas suas soluções foram (1,0) ou (0,1), extremos, não convergiu no ZDT4 e o espaçamento médio entre as soluções ficaram 1047071.1 muito maior que 87.466e-20 do G-MOPSO2, entre outros problemas de espaçamento. Não convergiu no FON, e teve hipervolume de apenas 113.65908. Uma maneira para lidar com a distância de aglomeração é usar uma seleção não proporcional a alguma medida, como no caso da seleção por torneio, inclusive ela é usada também pelo NSGA2 e entre outros [38].

O terceiro ponto é em relação ao L-MOPSO especificamente. As soluções não-dominadas, que são as melhores, são transmitidas de um agente a outro, e como cada agente tem um arquivo independente do outro, os arquivos terão soluções exatamente iguais entre si degradando a diversidade e podendo chegar a convergência prematura. Este problema é superado com o uso de um arquivo de pareto para população inteira, como no G-MOPSO e G-MOPSO2.

O quarto ponto diz respeito a melhor posição individual. Como a comparação é somente pela não dominância, uma vez que o agente encontrou uma solução ótima global, a melhor solução individual nunca será mais atualizada degradando a diversidade. O melhor seria se o agente atualizasse a melhor posição individual também pela densidade, em caso de a melhor solução individual e a atual serem mutuamente não dominadas.

O último ponto e o mais importante foi feito baseado numa análise de Xiaodong Li [42]. Para retratar o problema, é mostrado um exemplo na imagem a seguir:



Baseado no argumento deste pesquisador, mostra-se que o processo de atualização da melhor posição individual, apesar de ser eficaz para um objetivo, é ineficaz para comparações de dominância de pareto. Supondo que exista dois agente na posição atual $x_1$ e $x_2$, com a melhor posição individual $y_1$ e $y_2$, respectivamente e que a otimização seja de minimização. A figura mostra que $x_1$ e $y_1$ são mutuamente não dominados assim como $x_2$ em relação a $y_2$.

No entanto, $y_1$ é dominado por $x_2$ e $y_2$. E $x_1$ é dominado por $x_2$. Por ordenação de pareto, as duas melhores soluções são $x_2$ e $y_2$. A posição $x_2$ é melhor do que $y_2$, já que ela não é dominada por ninguém e ainda domina duas outras soluções. A solução $x_1$ é melhor que do que $y_1$, já que a primeira é dominada por um e a segunda por duas, ambas não dominam nenhuma solução. Logo, o que era de se esperar é que a melhor posição individual do agente um passasse de $y_1$ para $x_1$ e do agente dois de $y_2$ para $x_2$. Mas isto não acontece no PSO padrão, porque a comparação é feita localmente, é binária, sem tomar conhecimento da população inteira. Recordando o G-MOPSO e L-MOPSO, simplesmente o $x_1$ é comparado com o $y_1$, como o primeiro não domina o segundo a melhor posição individual não seria atualizada. O mesmo acontece com o agente dois.

Uma forma de superar este problema, é fazer uma ordenação de pareto em relação a toda população usando a posição atual e a melhor individual. Assim, os elementos da união dos conjuntos $x$ e $y$ são ranqueadas, fazendo o uso da informação da população em vez da local. O G-MOPSO2 faz uso dessa estratégia.

O resultado disso para o exemplo anterior, é que o G-MOPSO2 atualizaria a melhor solução para $x_1$ e $x_2$, enquanto os outros dois algoritmos ficariam estagnados no $y_1$ e $y_2$. De ante de tudo que foi visto, observa-se que a estratégia de convergência do G-MOPSO e L-MOPSO é imprecisa.

### 3.4 Dificuldades e Limitações

Ao observar os pseudocódigos, pode-se pensar que o desenvolvimento dos algoritmos é rápido e fácil, mas não é. A construção deles é exaustiva e penosa, e muitas vezes com erros que muitas vezes são encontrados *a posteriori.* Por isto, o último algoritmo criado(G-MOPSO2), foi escrito em Python em vez de C a fim de programar de forma mais rápida, enxuta e sem ficar entrando em pequenos detalhes de programação. Além disso, python é uma linguagem mais flexível do que C, e isto ajuda quando se quer modificar algoritmos. E juntamente com várias bibliotecas já prontas para gerar gráficos e estatísticas de forma fácil.

Outra limitação foi a dificuldade em encontrar a equação que gera o pareto front dos diversos *benchmarks.* Simplesmente, nas diversas literaturas pesquisada pelo autor, isto não foi divulgado, mas apenas o conjunto de pareto. O método para contornar isto seria a partir do conjunto de pareto, gerar vários pontos no espaço e determinar por interpolação polinomial a

equação do pareto front. No entanto, pode ser que a equação seja descontínua como no caso do ZDT3, o autor não sabe se tal método conseguiria resolver tal problema. A equação da geração do pareto front é importante porque a partir dela pode-se determinar as métricas de convergência, incluindo a mais usada que é do inglês *general distance.*

O autor acredita que *general distance* é melhor usado para determinar a convergência do que a taxa de acerto. Afinal, o primeiro determina a distância das soluções encontradas ao pareto front, enquanto o segundo só determina se as soluções estão ou não no front, nada diz a respeito do quão próximas elas estão. Inclusive, nos artigos observados o primeiro é usado e é o único usado para determinar a convergência.

Outro aspecto que pode ser considerado é, em vez de criar uma meta-heurística que procura resolver todos os tipos de problemas, é mais assertivo desenvolver uma que seja especializada em um tipo de problema, como linear, inteiro, não linear não-convexo, combinatorial e etc.

# Capítulo 4: Considerações

## 4.1 Contribuições

No aspecto teórico este trabalho me proporcionou conhecimento na área de otimização multi-objetiva, a forma como os se lida com otimização multi-modal, o conceito de pareto, a forma como tentam aumentar a variabilidade das soluções e as diversas técnicas de análise para comparar os algoritmos, que são as mais importantes. Afinal, os algoritmos mudam com o passar do tempo, mas a parte teórica, as idéias, a parte matemática por trás do tópico já andam a passos mais lentos. Enquanto a ideia de utilizar a evolução, algoritmos bio-inspirados baseados em população para otimização foram pensadas na década de cinquenta, inumeráveis tecnologias(algoritmos) foram pensados desde então. Outro aspecto teórico foi em estatísticas, pois foi necessário usar métricas para analisar e comparar os dados e os algoritmos, interpretando estas métricas e fazendo inferências em relação aos resultados.

Na parte tecnológica, aprendi a programar em python e usar suas bibliotecas para uso científico. E no aprendizado que python é melhor que C, para prototipação rápida, modificação de algoritmos e em testar as idéias de forma rápida. Além disso, aprendi sobre algoritmos tanto de otimização de um objectivo quanto para vários, como o algoritmo genético, programação genética, otimização por enxame de partículas e entre outros.

## 4.2 Trabalhos Futuros

Possivelmente irei estudar outras otimizações bio-inspiradas, estudar otimização multi-objetiva, otimização não linear e sucintamente processos estocásticos. Farei futuramente mestrado.

**Referências**

[1] Corne, David; Dorigo, Marco; Glover, Fred. New Ideas in Optimization, editora McGraw-Hill, Reino Unido, 1999.

[2] Bianchi, Leonora; Marco Dorigo; Luca Maria Gambardella; Gutjahr, Walter. "A survey on metaheuristics for stochastic combinatorial optimization". Natural Computing: an international journal, v. 8, p. 239–287, 2009.

[3] C. A., Coello"Evolutionary Multi-Objective Optimisation: A historical View of the Field", IEEE Computational Intelligence Magazine, vol 1, p. 28-36, 2006.

[4] Coello, Coello. "Twenty Years of Evolutionary Multi-Objective Optimization: A Historical View of the Field", 2005.

[5] Morse, Philip M; George, E."Operational Research in the British Army 1939–1945''.Methods of Operations Research, editora MIT Press & J Wiley, Reino Unido, 1954.

[6]Simon, Dan. "Evolutionary Optimization Algorithms: Biologically-Inspired and Population-Based Approaches to Computer Intelligence", editora Wiley, p. 3, 2013.

[7] Darwin, Charles. "On the Origin of Species by Means of Natural Selection", Londres, 1859.

[8] Mayr, Ernst. "What Evolution Is", Londres, editora Phoenix, 2002.

[9] Huxley, Julian. Evolution: The Modern Synthesis, editora MIT Press, Cambridge, 2010.

[10] K., Deb; A., Pratap; S., Agarwal; T., Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, Vol 6, No. 2, 2002.

[11] Coello, Coello; M.S., Lechuga. "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization". Proceedings of the IEEE Congress on Evolutionary Computation, vol 2, p. 1051–1056, 2002.

[12] Engelbrecht, Andries . Computaional Intelligence: An Introduction (2st ed.), editora Wiley, p. 127.

[13] Engelbrecht, Andries. Computaional Intelligence: An Introduction (2st ed.), editora Wiley, p. 128.

[14] V., Ramos; C., Fernandes; A.C., Rosa. “Social Cognitive Maps, Swarm Collective Perception and Distributed Search on Dynamic Land-scapes”. Technical report, Instituto Superior Técnico, Lisboa, Portugal, http://alfa.ist.utl.pt/~cvrm/staff/vramos/ref 58.html, 2005.

[15] J. Hoffmeyer. “The Swarming Body”. In I. Rauch e G.F. Carr, “Semiotics Around the World, Proceedings of the Fifth Congress of the International Association for Semiotic Studies”, p. 937–940, 1994.

[16] J. Kennedy eR.C. Eberhart. “Particle Swarm Optimization”. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1942–1948, 1995.

[17] Simon, Dan. “Evolutionary Optimization Algorithm: Biologically Inspired and Population-Based Approaches to Computer Intelligence”, p. 518, 2007

[18] Simon, Dan. “Evolutionary Optimization Algorithm: Biologically Inspired and Population-Based Approaches to Computer Intelligence”, p. 517, 2007

[19] Ehrgott, Matthias. “Multi-Criteria Optimization”, editora Springer, 2005.

[20] Deb, K. “Multi-objective optimization using evolutionary algorithms”, editora Wiley; 2001. p. 24, Reino Unido, 2001

[21] Zitzer, E.; Laumanns, M.; Thiele, L. “SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization”. In: Proceedings of the EROGEN Conference, p. 182–197, 2007.

[22] D.A., Van Veldhuizen; G.B., Lamont. “On measuring multiobjective evolutionary algorithm performance”. Proc. IEEE Congress Evolutionary, p. 204–211, 2000.

[23] E. Zitzler; L. Thiele. “Multiobjective optimization using evolutionary algorithms–A comparative case study”. Proc. Parall. Problem Solv. Nature, pp 292–301, editora Springer, 1998.

[24] E. Zitzler; L. Thiele. “Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach”, IEEE Trans. Evol. Comput., pp 257–271, 1999.

[25] J. R. Shcott. “Fault Tolerant Design Using Single and Multi-Objective Genetic Algorithms. Master’s Thesis”, Departament of Aeronautics and Astronautics, 1995.

[26] Bäck, Thomas. “*Evolutionary Algorithms in Theory and Practice”*, editora Oxford Univ. Press, p. 120, 1996.

[27] Miller, Brad; Goldberg, David (1995). "Genetic Algorithms, Tournament Selection, and the Effects of Noise". *Complex Systems* 9: p 193–212.

[28] M. Dorigo. “*Optimization, Learning and Natural Algorithms”*, PhD thesis, Politecnico di Milano, Italy, 1992.

[29] Shi, Y.; Eberhart, R.C.. "A modified particle swarm optimizer". *Proceedings of IEEE International Conference on Evolutionary Computation*. p. 69–73, 1998.

[30] WIKIPÉDIA. Desenvolvido pela Wikimedia Foundation. Apresenta conteúdo enciclopédico. Disponível em: https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-41. Acesso em: 28 Abril 2016.

[31] WIKIPÉDIA. Desenvolvido pela Wikimedia Foundation. Apresenta conteúdo enciclopédico. Disponível em: https://en.wikipedia.org/wiki/Marco_Dorigo. Acesso em: 28 Abril 2016.

[32] Dong Ping, Tian. “A Review of Convergence Analysis of Particle Swarm Optimization”, International Journal of Grid and Distributed Computing, vol 6, p. 117-128, 2013.

[33] ETH-ZURICH. Desenvolvido pelo Departamento de Tecnologia de Informação e Engenharia Elétrica. Apresenta o conteúdo. Disponível em: http://people.ee.ethz.ch/~sop/download/supplementary/testproblems/ . Acesso em: 28 Abril 2016.

[34] Kalyanmoy, Deb. “Multi-objective optmization using Evolutionary Algorithm”, editora Wiley, p. 36-37.

[35] Kalyanmoy, Deb. “Multi-objective optmization using Evolutionary Algorithm”, editora Wiley, p.42-43.

[36] E. Zitzler; L. Thiele; D. Deb. “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”, Evolutionary Computation, vol. 8, p. 177-178 ,2000.

[37] E. Zitzler; L. Thiele; D. Deb. “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”, Evolutionary Computation, vol. 8, p. 181-188, 2000.

[38] Kalyanmoy, Deb; Amrit, Pratap; Sameer, Agarwal; T., Meyarivan. “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”, IEEE Transactions on Evolutionary Computation, vol. 6, p. 182-197, Abril 2002.

[39] K., Deb; L. Thiele; M., Laumanns; E., Zitzler. “Scalable Test Problems for Evolutionary Multi-Objective Optimization.” Kanpur, India: Kanpur Genetic Algorithms Lab. (KanGAL), Indian Inst. Technol., 2001.

[40] E., Zitzler; M., Laumanns; L., Thiele. “SPEA2: Improving the Strength Pareto Evolutionary Algorithm.” Technical Report TIK-Report 103, Swiss Federal Institute of Technology Zurich (ETH), Maio 2001.

[41] Fonseca, C. M. and Fleming, P. J. . An overview of evolutionary algorithms in multi-objective optimization. Evolutionary Computation Journal, vol 3, 1995.

[42] Xiaodong, Li. “Genetic and Evolutionary Computation - GECCO”, p. 37-48, 2003.

[43] WIKIPÉDIA. Desenvolvido pela Wikimedia Foundation. Apresenta conteúdo enciclopédico. Disponível em: https://en.wikipedia.org/wiki/Self-organization . Acesso em: 11 de Maio 2016.

[44] Heylinghen, Francis. “The Science of Self-Organization System and Adptability”. Free University of Brussel, Belgium. Disponível em: http://pespmc1.vub.ac.be/papers/EOLSS-Self-Organiz.pdf . Acesso em: 11 de Maio 2016.

[45] WIKIPÉDIA. Desenvolvido pela Wikimedia Foundation. Apresenta conteúdo enciclopédico. Disponível em: https://en.wikipedia.org/wiki/Self-organization#Examples . Acesso em: 11 de Maio 2016.