Carlos Augusto Ligmanowski Junior

AVALIAÇÃO DE OTIMIZAÇÃO POR MÚLTIPLOS ENXAMES DE PARTÍCULAS NA SOLUÇÃO DE PROBLEMAS DINÂMICOS

Trabalho de Conclusão de Curso apresentado à banca examinadora da Universidade Federal do Paraná como exigência para obtenção do título de Bacharel em Ciência da Computação..

Área de concentração: *Inteligência Artificial*.

Orientador: Prof. Dr. Eduardo Jaques Spinosa .

Curitiba PR
2016

Resumo

Será apresentado nesse trabalho o algoritmo Multi-PSO, um algoritmo de multi-população para resolver problemas de otimização dinâmica. O Moving Peaks Benchmark da biblioteca DEAP do Python foi usado para simular esses problemas. Alguns de seus parâmetros e do Multi-PSO foram analisados. Para fazer essa analise, foram gerados gráficos que contém informações da melhor partícula durante a execução do algoritmo. Cada execução do algoritmo teve uma configuração diferente.

O trabalho tem como objetivo mostrar com o algoritmo de multi-enxame se comporta e quais características impactam no seu desempenho para resolver problemas de otimização dinâmica. É útil para quem se interessa pela área de inteligência artificial e deseja saber um pouco mais sobre otimização dinâmica , o MPB, PSO e Multi-PSO.

Palavras-chave: Algoritmo de multi-população, Biblioteca DEAP, Inteligência Artificial, Moving Peaks Benchmark, Multi-PSO, PSO, Parâmetros, Otimização dinâmica.

Abstract

The Multi-PSO algorithm will be presented in this graduation paper. It is a multi-population algorithm that is used in dynamic optimization problems. The Moving Peaks Benchmark(MPB), implemented on the DEAP framework in Python, was used to emulate dynamic problems. Some of the MPB attributes and Multi-PSO have been analyzed. Charts have been used to do this analysis and it contains the behavior of the best particle during the algorithm run. Each run was set with a diferrent setting.

This work has the main purpose of demonstrating how the multi-swarm algorithm behaves and what attributes have an impact in its performance on the MPB. It is useful for those who are interested in the Artificial Intelligence field and wish to know a little more about dynamic optimization, the MPB, PSO and Multi-PSO.

Keywords: Artificial Inteligence, Attributes, DEAP framework, Dynamic Optimization, Multi-PSO, Moving Peaks Benchmark, PSO.

Sumário

1 Introdução **1**

2 Otimização Dinâmica **3**

3 Otimização por enxame de partículas(PSO) **4**

3.1 Abordagem geral PSO 4

3.1.1 O Algoritmo PSO 5

3.2 PSO em cenários dinâmicos: o problema da falta de diversidade , mudanças que podem ser feitas no algoritmo básico para aumentar ou manter a diversidade . . 6

3.2.1 O problema da memória desatualizada 6

3.2.2 Aumentando a diversidade quando necessário 6

3.2.3 Mantendo a diversidade 8

3.3 Abordagem de multiplos-enxames (*multi-swarms*) 9

3.3.1 Mais sobre a Analogia de átomo 10

3.3.2 Exclusão 10

3.3.3 Anti-convergência 10

3.3.4 O algoritmo de Multi-Enxames (*multi-swarms*) 11

4 Estudo de caso: O Benchmark Moving Peaks (MPB) **14**

4.1 Descrição geral 14

4.2 Parâmetros 14

5 Experimentos **16**

5.1 Metodologia experimental 16

5.2 Impacto dos parâmetros 16

5.2.1 Número de picos 16

5.2.2 Severidade da mudança 19

5.2.3 Período da mudança 21

5.2.4 Correlação entre as mudanças sucessivas 23

5.2.5 Número de dimensões no espaço de busca 25

5.3 Características do algoritmo 27

5.3.1 A Anti-convergência 27
5.3.2 Número de partículas 28

6 Conclusões e Trabalhos Futuros **30**

Referências Bibliográficas **32**

Lista de Figuras

3.1 *Mais informações:* [Blackwell,] 7

3.2 A analogia de átomo. Um sub-enxame PSO de partículas neutras(círculos pintados) convergindo em um ótimo. O diâmetro do enxame neutro, $S0$, está diminuindo em um fator de $0,92$ em cada interação. Esse sub-enxame é cercado por partículas carregadas com diversidade constante $S-$. *Mais informações:* [Blackwell,] 9

5.1 Execução com 10 picos 17

5.2 Execução com 50 picos 18

5.3 Execução com 150 picos 18

5.4 Suave 20

5.5 Média 20

5.6 Severa 21

5.7 Período 1000 22

5.8 Período 20000 22

5.9 $Lambda_ = 0$ 24

5.10 $Lambda_ = 0.5$ 24

5.11 $Lambda_ = 1$ 25

5.12 5 dimensões 26

5.13 15 dimensões 26

5.14 5 partículas 28

5.15 15 partículas 29

Lista de Tabelas

5.1 Valores dos parâmetros para teste de severidade da mudança no ambiente . . . 19
5.2 A tabela mostra a execução do algoritmo para o número de picos p=10 e p=200. Os números são a média dos erros da melhor partícula. Dados retirados de [Blackwell,] 27

Lista de Acrônimos

DEAP	Distributed Evolutionary Algorithms in Python
EAs	Evolutionary Algorithms
MPB	Moving Peaks Benchmark
MULTI-PSO	Algoritmo Multiple Particle Swarm Optimization
PSO	Algoritmo Particle Swarm Optimization
UFPR	Universidade Federal do Paraná

Capítulo 1

Introdução

A Ciência da Computação trouxe e continua a trazer uma revolução na maneira em como as pessoas vivem, interagem, criam e trabalham. Isso se deve ao fato de que os computadores são capazes de realizar tarefas que demandavam muito esforço e tempo em tempos quase instantâneos. Uma área da Computação que desde o principio foi bastante estudada é a de problemas de otimização. Desde tarefas do dia a dia, como problemas complexos de decisão critica, são problemas que a humanidade quer otimizar. Ainda assim, alguns problemas são muito difíceis de resolver, ou até impossíveis atualmente, mesmo para computadores. Resolver aqui significa encontrar a melhor solução. Então, uma área começou a se desenvolver bastante dentro da computação, a Inteligência Artificial.

Nela, problemas de otimização dinâmica são bem comuns, que são representantes de problemas do mundo real. Os problemas de otimização são modelados por funções matemáticas e otimizar esses problemas significa encontrar uma solução ótima para os mesmos, não necessariamente a melhor de todas. Esse trabalho foca em uma parte da Inteligência Artificial que é a Computação Evolutiva. Mais especificamente no algoritmo de otimização dinâmica Multi-PSO.

Primeiro, é apresentado o algoritmo canônico PSO e sua evolução, o Multi-PSO. Ao compreender a evolução dos algoritmos e como funcionam, o ambiente em que são aplicados é introduzido. Então, o *Moving Peaks Benchmark(MPB)* é introduzido, o qual é usado para simular problemas reais e é muito usado por pesquisadores na área de Otimização Dinâmica. Após o MPB ser analisado, aplica-se o algoritmo Multi-PSO no ambiente do MPB e durante as execuções alguns dos parâmetros do MPB são alterados para verificar seu impacto no desempenho do algoritmo. Por fim, alguns parâmetros do próprio algoritmo são testados para verificar sua eficiência.

Depois da fase de experimentos, relata-se o que foi observado neles e suas conclusões são expostas. Entretanto no Capítulo 2 é apresentada um pouco sobre como funciona a otimização dinâmica. Por outro lado, já no Capitulo 3, o algoritmo PSO e Multi-PSO são definidos. Os parâmetros do MPB, e a biblioteca DEAP[Branke,] são mostrados no próximo Capitulo 4. Contudo no Capítulo 5 é explicada a metodologia experimental aplicada para avaliar os

parâmetros do MPB e do Multi-PSO. E por fim, porém não menos importante é exposta a conclusão sobre os experimentos e ideias direcionando para trabalhos futuros.

Capítulo 2

Otimização Dinâmica

O conceito de otimização é: decisões que melhorem certo processo partindo de uma formulação matemática do problema[Goossens et al., 1990]. Otimização dinâmica refere-se ao problema que muda seus ótimos a medida que o tempo passa, ou seja, o nosso mundo real.

Muitos problemas de otimização do mundo real são sujeitos a mudar de condição por meio de sua existência, então, conseguir otimizar em um ambiente dinâmico é muito importante. Essas mudanças de condição alteram a função objetivo e a instância atual do problema. Exemplos desse tipo de problema são: chegada de novas tarefas em um sistema operacional , máquinas pararem de funcionar dentro de uma fabrica ou indústria, mudança no mercado financeiro e a mudança de recursos disponíveis em algum tipo de tarefa. Qualquer tipo de tarefa humana pode ser otimizada em certo aspecto.

Para ficar mais fácil de simular problemas desse tipo, *benchmarks* foram criados por pesquisadores para testarem seus algoritmos. Essa é uma área em constante desenvolvimento e o futuro será baseado em como certos algoritmos conseguem trazer soluções ótimas nas mais variadas situações e momentos.

Capítulo 3

Otimização por enxame de partículas(PSO)

Neste capitulo, o algoritmo PSO é apresentado, ele é o algoritmo base do Multi-PSO e seu funcionamento é baseado em movimento de partículas, que são variáveis que procuram por uma solução do problema enquanto se movimentam no ambiente.

3.1 Abordagem geral PSO

A otimização por enxame de partículas(PSO) é um algoritmo de otimização baseado em populações e em muitos aspectos é similar a algoritmos evolutivos (EAs), que são algoritmos baseados na evolução biológica que tem por objetivo a busca de uma solução para problemas de otimização. O algoritmo tem 2 componentes principais que são: as dinâmicas de partícula e a informação que as partículas compartilham dentro do enxame(*swarm*). As dinâmicas de partículas, que é o movimento delas para procurar uma solução ótima, são inspiradas em movimentos de grandes grupos de animais em bando feitos em simulações de computação gráfica[C.Reynolds, 1987]. As partículas do PSO não seguem uma trajetória constante. Ao invés disso, as partículas se movem em saltos, movimento conhecido como *flight*[Iskandar, 1983]. As informações compartilhadas são inspiradas por redes sociais[R.Eberhart e Y.Shi, 2001].

As partículas possuem: a melhor localização visitada até o momento(pbest), e seu *fitness*, que é o seu valor na função objetivo, e também, tem acesso à melhor localização geral de qualquer partícula do enxame de partículas. Essas 2 localizações se tornam *attractors*, ou seja, cada partícula vai ser repetidamente atraída para vizinhanças perto dessas 2 localizações. Se alguma localização melhor é encontrada, os valores de melhor são atualizados. Esse acesso à informações entre as partículas se dá ao fato de estarem em uma topologia de rede. Algumas topologias foram testadas e as topologias em estrela e *fully connected* acabam sendo escolhas populares. O algoritmo testado nesse trabalho usa uma topologia *fully connected*: toda partícula

compartilha informação com toda partícula. Assim é possível saber a partícula com a melhor localização(gbest).

As partículas também possuem velocidade, que influencia nas atualizações de posição. A velocidade e posição da partícula são calculados assim:

- $v(t+1) = v(t) + a(t+1)$ *Eq.1*
- $p(t+1) = p(t) + v(t+1)$ *Eq.2*

Onde a, v, p e t são respectivamente: aceleração, velocidade, posição e tempo(contador de iteração).

Além disso, cada partícula sofre uma atração por cada *attractor*. Essa atração é linear e é regulada randomicamente, por um número randômico uniformemente distribuído. A convergência para uma solução ótima não segue somente essas dinâmicas, o movimento da partícula(*flight*) deve também progressivamente se contrair. Essa contração é implementada Clerc e Kennedy com um fator de constrição χ, $\chi < 1$[Clerc e Kennedy, 2000]. O algoritmo do Clerc e Kennedy é considerado aqui como o algoritmo PSO canônico. Então, a aceleração da partícula na *Eq.1* é dada por:

- $ai = \chi[c\varrho.(pg - xi) + c\varrho.(pi - xi)] - (1 - \chi)vi$ *Eq.3*

onde ϱ são vetores de números randômicos gerados em uma distribuição uniforme U[0,1], c é chamada *spring constant* e tem valor $c > 2$. pi e pg são os *attractors*, as melhores posições de partícula e global até o momento. Clerc e Kennedy derivam uma relação para o fator de constrição $\chi(c)$: valores padrão são $c = 2,05$ e $\chi = 0,729843788$[Clerc e Kennedy, 2000].

3.1.1 O Algoritmo PSO

```
1  PARA CADA partícula i {
2        iniciar randomicamente vi, xi
3        pi = xi
4        Avaliar( f(pi) )
5        pg = argmax f(pi)
6  }
7
8  REPETIR {
9        PARA CADA partícula i {
10             Atualizar(xi) // Atualizar posição da partícula de acordo com as
11 equações Eq.1 , Eq.2 e Eq.3
12             Avaliar( f(xi))
13             SE f(xi) > f(pi) ENTÃO {
14                   pi = xi
15                   SE f(xi) > f(pg) ENTÃO {
16                         pg = xi
```

```
17                      }
18                  }
19  } ATÉ um critério de parada ser alcançado
20  FIM
```

3.2 PSO em cenários dinâmicos: o problema da falta de diversidade , mudanças que podem ser feitas no algoritmo básico para aumentar ou manter a diversidade

A aplicação do PSO em problemas dinâmicos é explorada por vários autores. A consequência geral desses estudos é que o PSO canônico de Clerc e Kennedy deve ser modificado para se obter os melhores resultados em ambientes do Moving Peaks Benchmark. A origem dessa dificuldade está em 2 problemas: memória desatualizada(*outdated memory*) por causa da mudança que ocorre nos cenários e perda de diversidade(*diversity loss*) por causa da convergência que acontece para certos ótimos. Desses 2 problemas, a perda de diversidade é , de longe, o mais sério.

3.2.1 O problema da memória desatualizada

Memória desatualizada ocorre quando acontece uma mudança no ambiente ou cenário. O ótimo altera-se em localização ou valor. A memória da partícula (melhor localização visitada até agora e seu *fitness*) torna-se falsa em um outro cenário, com efeitos desastrosos no desempenho do PSO.

O problema de memória desatualizada é resolvido ao calcular o *fitness* da melhor localização em todas as iterações , porque isso pode detectar mudanças no cenário se os valores não forem iguais. O efeito disso é a duplicação das avaliações por iteração.

3.2.2 Aumentando a diversidade quando necessário

Foi demonstrado que o tempo que leva para um enxame parcialmente convergido se diversificar e convergir de novo quando ocorre uma mudança é grande e muito prejudicial ao desempenho do algoritmo. Então , precisa-se de mecanismos para lidar com isso: diversificação quando ocorre uma mudança no ambiente ou uma maneira de manter a diversidade durante a execução do algoritmo. Existem 4 mecanismos para diversificação ou para manter a diversidade: randomização, repulsão, redes dinâmicas e multi-populações.

O problema da diversidade perdida

Uma diversidade insuficiente, quando há uma mudança, faz com que seja muito difícil o enxame detectar um ótimo porque como dito, a população do enxame leva tempo para se diversificar e convergir em um novo ótimo. Um conceito que ajuda a entender a dificuldade da perda de diversidade, é o diâmetro do enxame |S|. É a maior distância em qualquer direção e entre quaisquer 2 partículas[Blackwell, 2003b]. Essa distância pode ser usada como medida de diversidade no enxame (Fig. 3.1) .

Quando uma mudança ocorre, a localização do novo ótimo pode estar dentro ou fora do enxame. Se o novo ótimo estiver significativamente longe do enxame, as baixas velocidades das partículas vão inibir uma diversificação ou mesmo tornar muito difícil encontrar o ótimo. O enxame pode ainda oscilar baseada em um *attractor* falso e por uma linha perpendicular ao verdadeiro ótimo , em um fenômeno conhecido como colapso linear(*linear colapse*).

Existem 2 soluções na literatura para o problema de diversidade insuficiente ou diversidade perdida: ou um mecanismo de aumento de diversidade pode ser usado quando tem uma mudança no ambiente(ou também em intervalos pré estabelecidos) ou manter a diversidade sempre durante a execução do programa[Blackwell e Branke,].

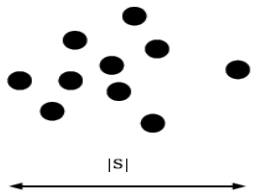

Figura 3.1: *Mais informações:* [Blackwell,]

A Diversificação

Hu e Eberhart[Hu e Eberhart, 2002] estudam alguns mecanismos de diversificação do enxame de partículas. Todos eles envolvem randomização de tudo ou parte do enxame de

partículas. Isso acontece quando a reavaliação da função objetivo nos *attractors* detecta uma mudança.

Randomização implica em perda de informação, porque tem o perigo de se apagar muitas partículas e, com isso, gerar um enxame praticamente novo. Por outro lado, pouca randomização talvez não traga uma diversidade grande o suficiente para lidar com a mudança no cenário. Essa arbitrariedade , muito ou pouca randomização, só pode ser resolvida com muito conhecimento prévio sobre o dinamismo da função objetivo ou com algum mecanismo de modificação de alto nível implementado. Esse mecanismo poderia deduzir detalhes sobre o dinamismo da função, fazendo ajustes apropriados nos parâmetros de diversificação. Até agora, mecanismos de alto nível como esse ainda não foram estudados.

3.2.3 Mantendo a diversidade

A seguir, serão apresentados mecanismos para manter a diversidade e quais desses mecanismo o algoritmo desse trabalho, o Multi-PSO usa para resolver o problema da diversidade.

Mantendo a diversidade por repulsão

Um certo grau de diversidade, constante e bom o bastante, pode ser mantido através de algum tipo de mecanismo de repulsão. Um exemplo de repulsão que foi testado em um cenário dinâmico é a analogia de átomo[Blackwell e Bentley, 2002][Blackwell e Branke, 2004]. A analogia de átomo consiste de um enxame com partículas carregadas(“charged particles”) orbitando um núcleo de partículas neutras que usam o PSO (Fig. 3.2). Essas partículas carregadas aumentam a diversidade nas redondezas do enxame neutro. Qualquer mudança no ótimo dentro do enxame como um todo, deve ser detectável. Uma boa detecção, melhor que no PSO clássico, foi demonstrada por experimentos dinâmicos unimodais[Blackwell, 2003c].

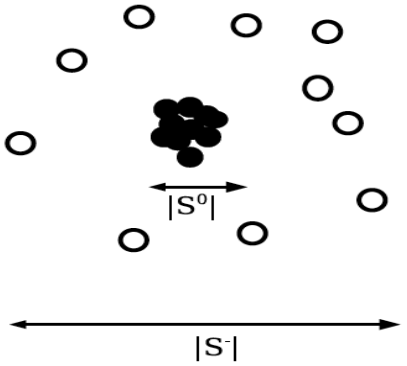

Figura 3.2: A analogia de átomo. Um sub-enxame PSO de partículas neutras(círculos pintados) convergindo em um ótimo. O diâmetro do enxame neutro, $S0$, está diminuindo em um fator de $0,92$ em cada interação. Esse sub-enxame é cercado por partículas carregadas com diversidade constante $S-$. *Mais informações:* [Blackwell,]

Mantendo a diversidade com multi-populações

Um número de grupos de pesquisa consideram multi-populações a melhor maneira de manter a diversidade[Blackwell,]. A ideia de multi-população é particularmente útil em ambientes com vários ótimos como o MPB, ou seja, em ambientes reais. O objetivo é fazer cada população convergir para um pico diferente ou ótimo diferente. Se, em algum momento, ocorrer uma mudança e outro pico se tornar o ótimo global, uma população estaria por perto para convergir sobre ele. Técnicas de multi-população incluem: *niching* , *speciation* e *multi-swarms*. A melhor delas para problemas dinâmicos é a de *multi-swarms* porque é a que tem melhores resultados em ambientes dinâmicos. Por esse motivo, multi-swarms vai ser o foco desse trabalho.

3.3 Abordagem de multiplos-enxames (*multi-swarms*)

Uma idéia boa seria combinar multi-populações com mecanismos para manter a diversidade nos enxames, como a repulsão. Essa idéia, chamada de multi-enxames, foi proposta por primeiro em [Blackwell, 2003a] em um contexto que não priorizava otimização. A extensão do multi-enxame para uma abordagem de otimização dinâmica foi feita por Blackwell e

Branke[Blackwell e Branke, 2004] e foi inspirada pelo algoritmo de sentinelas, o *self-organizing scouts (SOS)* do próprio Branke[Blackwell, 2001].

Um multi-enxame é uma colônia de enxames com partículas carregadas que interagem localmente por exclusão e globalmente por anti-convergência. Exclusão previne dois ou mais enxames de convergir em um mesmo ótimo e anti-convergência mantém a diversidade em todos os enxames como um todo. Então, os principais componentes do algoritmo de multi-enxame são: analogia de átomo, exclusão e anti-convergência.

3.3.1 Mais sobre a Analogia de átomo

Como anteriormente, cada enxame é visto como um átomo: um núcleo neutro com partículas se contraindo de acordo com o algoritmo PSO e com partículas carregas o envolvendo. Como no átomo, a exclusão mutua entre essas partículas carregadas seguem uma regra determinística e clássica, a repulsão de Coulomb.

Alternativamente, tem o átomo quântico, em que as partículas são posicionadas dentro de uma hiperesfera de raio r centrada em pg(attractor global) de acordo com uma distribuição de probabilidade. O átomo quântico tem a vantagem de menor complexidade e de ter uma distribuição facilmente controlável[Blackwell,].

3.3.2 Exclusão

A exclusão é inspirada no principio de exclusão da física: 2 elétrons não podem ter o mesmo estado em um mesmo momento. A versão deste principio para múltiplos-enxames é a regra que proíbe dois enxames de entrar no raio de exclusão um do outro, onde a distancia entre os dois enxames é definida como a distancia entre suas melhores partículas(pg).

O operador de exclusão simplesmente randomiza o pior enxame em qualquer colisão.

3.3.3 Anti-convergência

Anti-convergência é um simples operador que é feito para garantir que pelo menos um enxame está livre no múltiplo-enxame em qualquer momento. Um enxame livre é um enxame que patrulha o espaço de busca ao invés de convergir em um ótimo local ou global.

A idéia da anti-convergência é que se o número de enxames for menor que o número de ótimos, todos eles vão convergir, e assim deixam alguns ótimos sem ser verificados e, um desses ótimos, pode ser o ótimo global. A presença de enxames livres mantém a diversidade do multi enxame e também aumenta a resposta nas mudanças de cenário.

Nota-se que existem dois níveis de diversidade: diversidade no nível do enxame, reforçada pelo principio da exclusão, e a diversidade em nível geral do multi-enxame que possibilita o multi enxame, como um todo, procurar por novos ótimos ou picos.

A cardinalidade do multi-enxame

O número de enxames no algoritmo de multi-enxame pode ser analisado em relação ao número de ótimos ou picos no MPB. Espera-se que a anti-convergência seja benéfica quando o número de enxames for menor que o número de picos ou ótimos , $M < p$. Se $M > p$, a presença de enxames livres pode absorver algumas informações de alguns ótimos. O caso ideal seria $M = p$, onde o número de enxames seria igual ao número de picos. Nesse caso, a anti-convergência poderia ser desativada.

3.3.4 O algoritmo de Multi-Enxames (*multi-swarms*)

```
1  // Multi Enxame
2
3  //Inicialização
4  PARA CADA partícula ni { // n=swarm i=particula
5        iniciar randomicamente vi,xi
6        pi = xi
7        Avaliar(pi)
8  }
9
10 PARA CADA enxame n {
11       png = argmax{ f(pi) } // melhor partícula pi
12 }
13
14 // Fim da inicialização
15
16 REPETIR {
17       // Anti-Convergência
18       SE todos os enxames convergiram ENTÃO
19             Adicionar(novo_enxame)
20       SE numero_enxames > Nexcesso ENTÃO
21             Remover(pior_enxame)
22
23       PARA CADA enxame n {
24             // Testar mudança
25             Avaliar(png)
26             SE novo valor png édiferente da última iteração ENTÃO
27                   Reavaliar( todos os pi ) // \textit{attractors}
28                   Atualizar attractor do enxame - pg
29
30             PARA CADA particula i do enxame n {
31                   // Atualizar posição partícula
32                   Aplicar equações dependendo do tipo de partícula
33                   // Atualizar Attractor
34                   Avaliar(xni)
35                   SE f(xni) > f(pni) ENTÃO {
```

```
36                          pni = xni
37                          SE f(xni) > f(png) ENTÃO
38                                png = xni
39                    }
40              }
41
42        }
43
44        // Exclusão
45        PARA CADA enxame m != n {
46              SE png(attractor do enxame) esta dentro rexcl(raio de exclusão) de pmg ENT
47                    // Reinicia o pior enxame
48                    SE f(png) <= f(pmg) ENTÃO
49                          Reiniciar( n ) // Reinicia o enxame n
50                    SENAO
51                          Reiniciar( m ) // Reinicia o enxame m
52                    PARA CADA partícula no enxame reiniciado {
53                          Reavaliar xi e pi
54                          Atualizar attractor do enxame
55                    }
56              }
57        }
58
59  } ATÉ numero de avaliações > max_avaliacoes
60  FIM
```

Descrição do Algoritmo de Multi-Enxame

O algoritmo começa com a inicialização randômica de todas as partículas. Depois disso, o algoritmo verifica a anti-convergência ao verificar se todos os enxames convergiram. Se todos enxames convergiram , adiciona-se um enxame livre. E se existem mais enxames que um número permitido de enxames livres, chamado no algoritmo de NEXCESS, o pior de todos é removido do multi-enxame. Então, cada enxame é verificado.

Após o teste de convergência, verifica-se se houve mudança no cenário ao avaliar o *attractor* global png (n é o índice do enxame). Se png mudou desde a última iteração , então houve mudança no cenário e como já mencionado no problema de memória desatualizada, todos os *attractors* de todas as partículas do enxame são reavaliados.

A atualização do enxame acontece nesse momento. Todas as partículas mudam de posição e os *attractors* são atualizados.

O próximo teste realizado é o de exclusão. Caso um enxame esteja dentro do raio de exclusão de outro (rexcl), o pior dos dois enxames é reiniciado randomicamente e seus valores de attractors recalculados.

E assim , depois de todos os enxames serem verificados na etapa de testes, tudo é feito de novo até se alcançar um número de avaliações máximo.

Capítulo 4

Estudo de caso: O Benchmark Moving Peaks (MPB)

Nesse capitulo, fala-se mais sobre o MPB e como ele foi implementado. O objetivo é dar uma visão mais clara de como funciona e o que pode-se fazer com ele.

4.1 Descrição geral

Os problemas do mundo real geralmente possuem ótimos ou soluções que variam com o tempo. Para avaliar e comparar algoritmos de otimização dinâmica, pesquisadores criaram benchmarks.

Branke criou o MPB devido a necessidade de simular problemas reais. O MPB é uma função objetivo que muda com o tempo. Ela consiste de número de picos , que mudam em altura, largura e localização periodicamente[du Plessis e Engelbrech, 2013]. A função objetivo f(x,t) está otimizada nos picos.

O MPB usado nesse trabalho foi o implementado pela biblioteca DEAP para a linguagem Python[Branke,].

4.2 Parâmetros

O MPB possui os seguintes parâmetros que podem ser mudados:

- dim: Número de dimensões. É o tamanho do espaço de busca.
- pfunc: Função objetivo dos picos.
- npeaks: Número de picos. Se for um inteiro, o número de picos não muda, se for uma sequência de 3 números, os picos vão flutuar entre esses valores.
- min_coord: Coordenada mínima do centro dos picos

- max_coord: Coordenada máxima do centro dos picos
- min_height: Altura mínima dos picos, o *fitness* da função
- max_height: Altura máxima dos picos, o *fitness* da função
- uniform_height: Valor inicial de altura para todos os picos. Se esse valor é menor igual que 0, o valor é iniciado randomicamente para cada pico.
- min_width: Largura mínima dos picos
- max_width: Largura máxima dos picos
- uniform_width: Valor inicial de largura para todos os picos. Se esse valor é menor igual que 0, o valor é iniciado randomicamente para cada pico.
- lambda_: correlação entre as mudanças sucessivas dos picos
- move_severity: Distância que um pico se move quando uma mudança no ambiente ocorre
- height_severity: Desvio padrão da mudança feita pela altura do pico quando os picos mudam, ou seja, limite da mudança nos ótimos para mais ou menos.
- width_severity: Desvio padrão da mudança feita pela largura do pico quando os picos mudam.
- period: Número de avaliações até ocorrer uma mudança no cenário

Capítulo 5

Experimentos

Neste capitulo, alguns experimentos com os parâmetros do MPB e do algoritmo Multi-PSO foram feitos, para observar o impacto deles na resolução de problemas de otimização dinâmica. A seguir, mostra-se o que foi usado e os resultados dos experimentos.

5.1 Metodologia experimental

Foram feitos testes para avaliar o impacto dos parâmetros do MPB no algoritmo de Multi-Enxame aqui proposto[Blackwell,] e também a eficácia de alguns dos parâmetros do próprio algoritmo. Isso foi feito porque o objetivo principal do algoritmo é ser usado em problemas no mundo real, e sabendo alguns dos impactos de parâmetros, torna-se mais fácil de compreender alguns comportamentos que o algoritmo possa ter em tais problemas.

5.2 Impacto dos parâmetros

Nesta seção, discute-se sobre o impacto dos parametros do MPB no desempenho do algoritmo.

5.2.1 Número de picos

Os parâmetros principais utilizados para os testes do número de picos no algoritmo foram :

- Número de dimensões : 5
- Número de avaliações: 20000
- Número de partículas por enxame: 5
- Altura inicial randômica

- Período de mudança: 5000 avaliações

De maneira geral, o número de picos interfere no algoritmo principalmente nos momentos de mudança de cenário. Quanto maior o número de ótimos, mais severas tornam-se as mudanças. Esse é um ponto esperado porque os ótimos globais podem estar muito distantes um dos outros dependendo da mudança.

Abaixo, gráfico de algumas execuções:

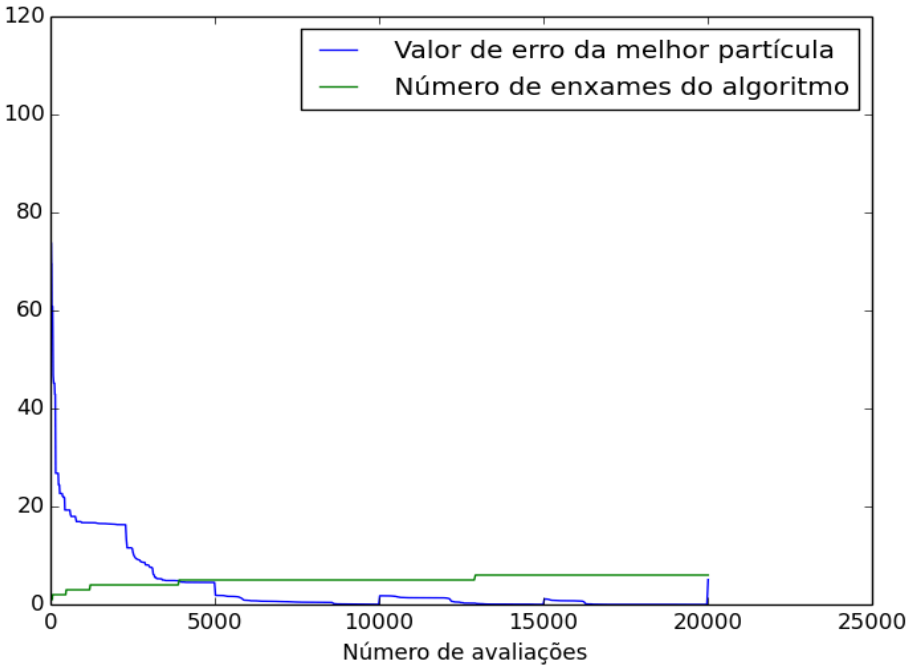

Figura 5.1: Execução com 10 picos

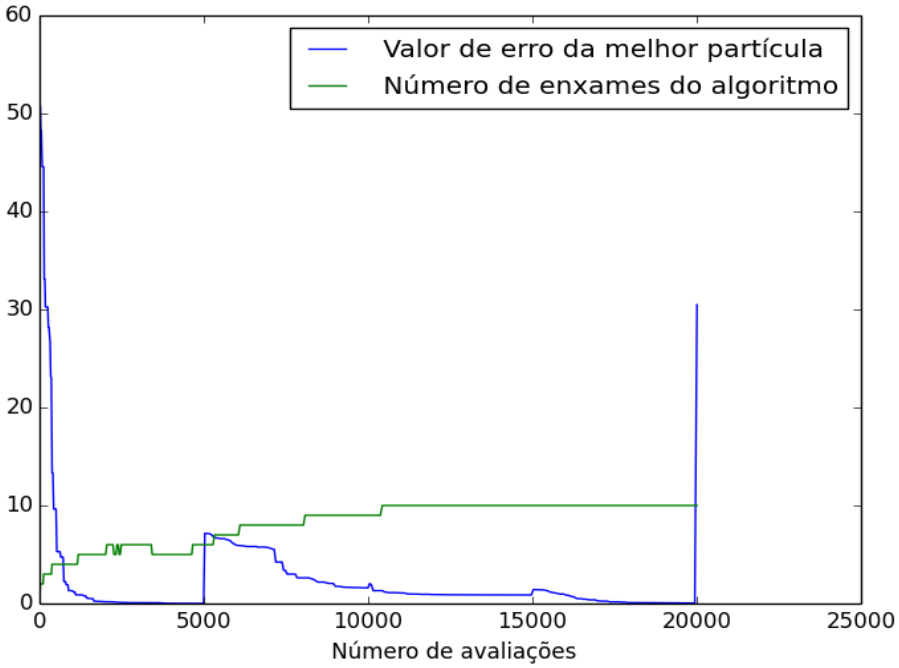

Figura 5.2: Execução com 50 picos

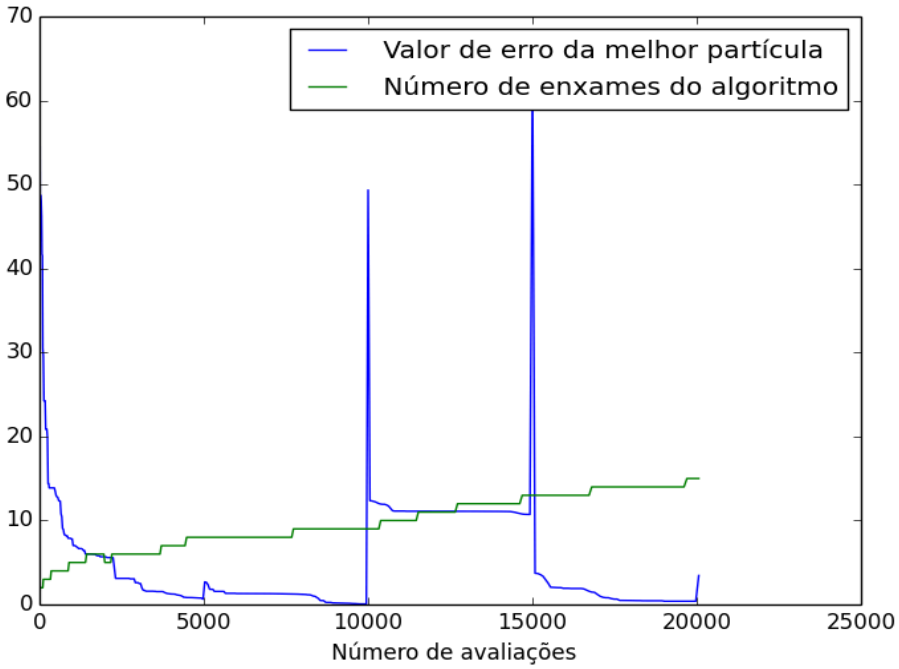

Figura 5.3: Execução com 150 picos

Verifica-se que a execução para 10 e 50 picos não possui diferenças significativas quanto ao desempenho(valor erro da melhor partícula) e nem no número de enxames criados . Um

ponto a ressaltar é que a cada 5000 avaliações ocorre uma mudança. Já com 150 picos, algumas mudanças tem um impacto severo no desempenho, como em 10000 e 15000 avaliações. E também, com 150 picos, a busca pelo ótimo global é mais lenta, porém, a medida que o algoritmo cria novos enxames, o desempenho começa a ficar próximo dos 10 e 50 picos. Isso é um exemplo da cardinalidade do multi enxame.

5.2.2 Severidade da mudança

A severidade na mudança diz respeito a 3 parâmetros do MPB: move_severity, height_severity e width_severity, explicados na seção 4. Os parâmetros foram mudados de acordo com a tabela:

Tabela 5.1: Valores dos parâmetros para teste de severidade da mudança no ambiente

	move_severity	height_severity	width_severity
Suave	1	1	0.01
Média	3	5	0.5
Severa	5	10	1

O cenário usado continua sendo o Cenário 2 do MPB[2]. Parâmetros significativos:

- min_height: 30
- max_height: 70
- min_width: 1
- max_width: 12
- Número de avaliações: 80000
- Número picos: 50
- Periodo de mudança: 5000 avaliações

A seguir, alguns gráficos das mudanças:

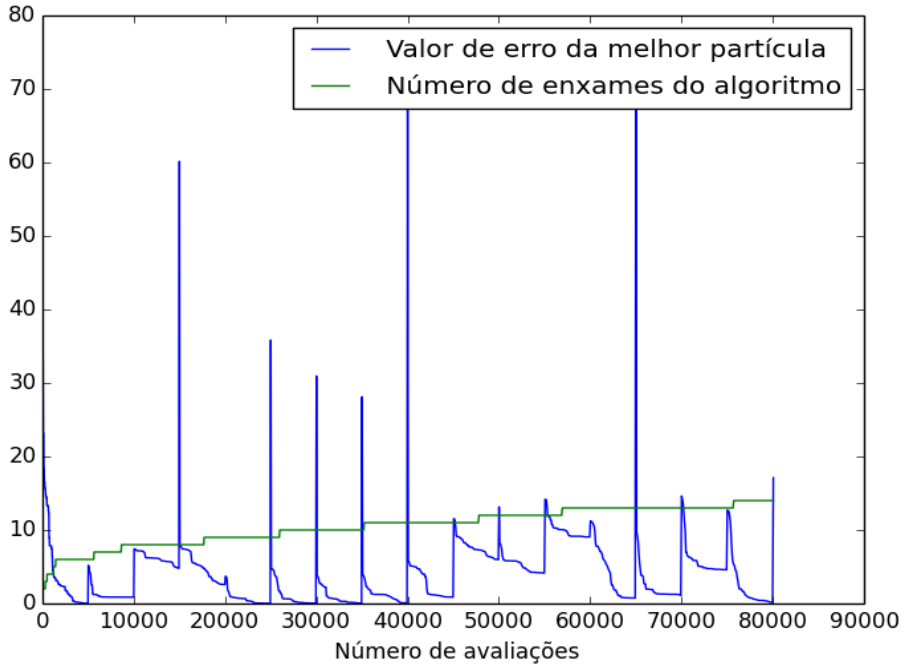

Figura 5.4: Suave

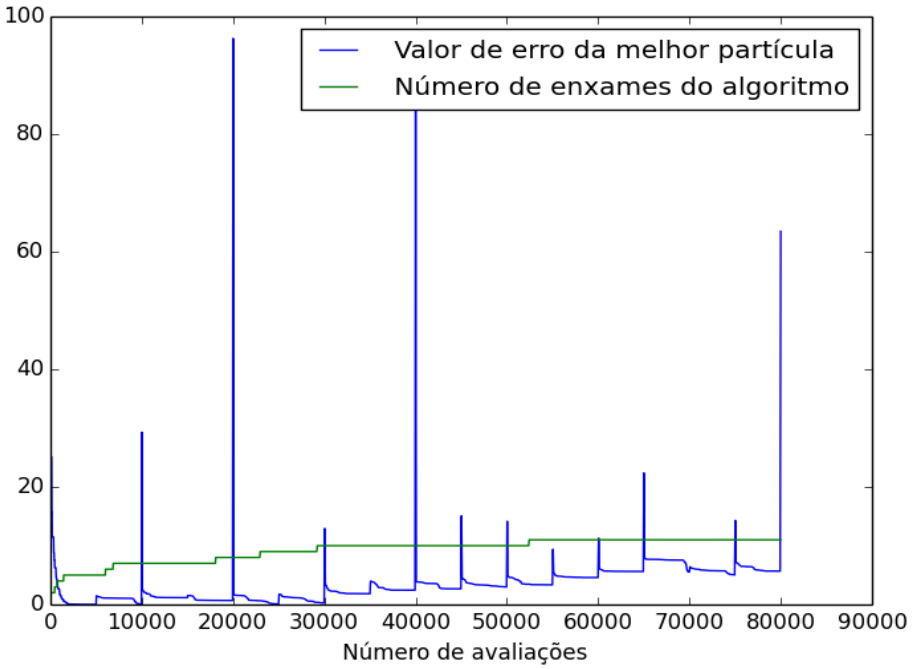

Figura 5.5: Média

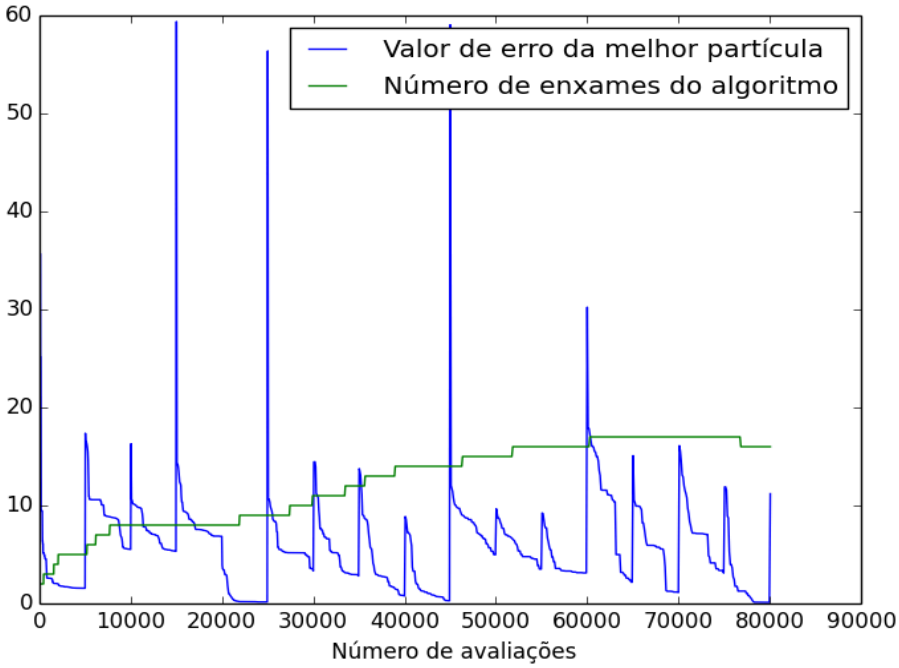

Figura 5.6: Severa

Observa-se que mudanças mais fortes impactam muito no desempenho do algoritmo. Isso pode ser visto nos gráficos. Na Fig. 5.4, as mudanças foram pouco intensas, com os ótimos depois das mudanças permanecendo por perto dos ótimos de antes dela. Isso faz com que a convergência dos enxames seja rápida para os novos picos logo depois das mudanças. Na Fig. 5.5 , as mudanças afetam mais o algoritmo, principalmente entre 45000 e 65000 avaliações , pois ali a mudança foi a mais drástica em todo o ambiente. Percebe-se que o algoritmo demora para convergir em ótimos globais nesse período. Já na Fig.5.6, existem vários momentos de mudança drástica. Com o tempo o algoritmo consegue convergir mais rápido por causa do maior número de enxames, porém, o impacto no desempenho na mudança severa é bem significativo e seria uma área onde podem haver otimizações no algoritmo.

5.2.3 Período da mudança

Problemas do mundo real estão sujeitos a mudanças periódicas. O *Moving Peaks Benchmark* tem o parâmetro *period* que se refere ao número de avaliações de particula e sempre que chegar nesse número, ocorre uma mudança no ambiente. Assim, foram feitos testes focando na mudança desse parâmetro e no que ele influência no desempenho do algoritmo.

Os testes foram feitos no cenário 2 com os seguintes parâmetros mais significativos:

- Número de picos: 50
- Períodos: 1000 e 20000 avaliações

A seguir, os gráficos com os períodos:

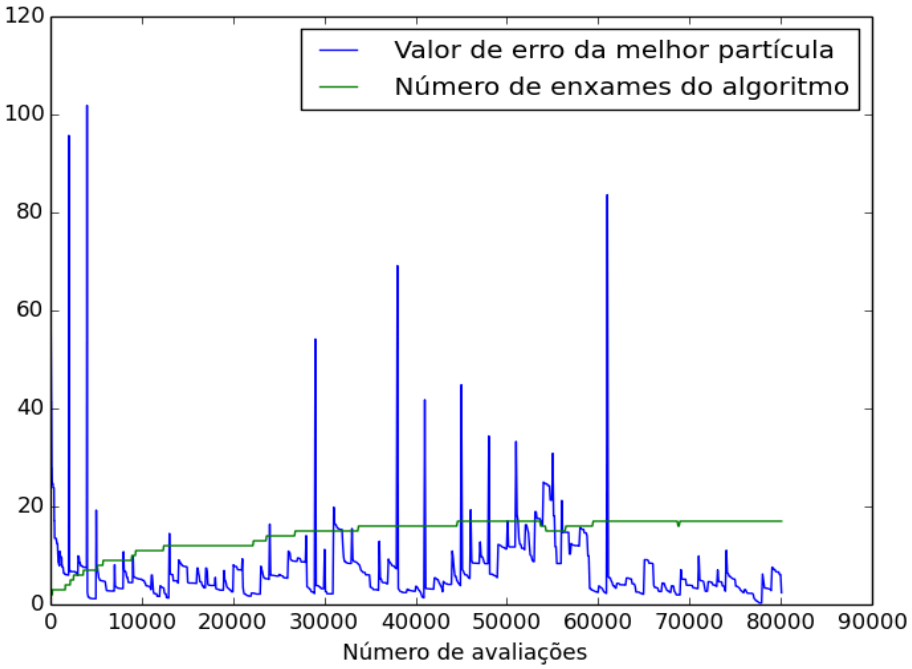

Figura 5.7: Período 1000

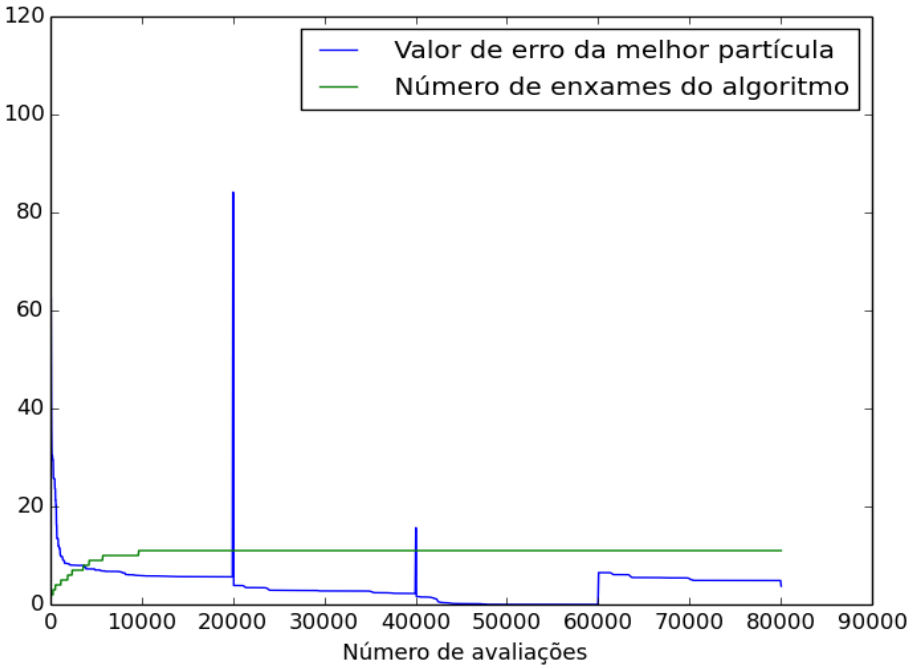

Figura 5.8: Período 20000

Percebe-se um ambiente extremamente caótico na Fig. 5.7, pois não há tempo para convergir em um ótimo global antes de uma mudança. Este tipo de ambiente não parece muito bom para o algoritmo de multi enxame quando se tem poucos enxames. A medida que o número de enxames aumenta, as mudanças parecem ter menos impacto no desempenho.

Na Fig. 5.8, o ambiente esta mais em equilíbrio pois ocorrem menos mudanças. Isso é muito bom para o algoritmo multi-enxame e ele converge rapidamente ao ótimo global.

Ambientes do mundo real, diferente do benchmark, são ambientes onde o período de mudança são variáveis, com a possibilidade de se alternar em caos e equilíbrio. Baseando-se nos testes, o algoritmo teria dificuldades em ambientes muito caóticos com número variado de ótimos. Seria uma parte boa para analisar otimizações.

Uma coisa interessante ocorre na Fig. 5.8 que é um momento onde o algoritmo fica preso em um ótimo local, mesmo com a anti-convergência e exclusão.

5.2.4 Correlação entre as mudanças sucessivas

A correlação refere-se ao parâmetro *lambda_* do MPB e tem valores entre 0 e 1, que medem o nível de dependência entre as mudanças sucessivas no ambiente. Foram feitos testes no cenário 2 com os seguintes parâmetros significativos:

- Número de picos: 50
- Período: 5000
- Numero de avaliações: 20000
- Correlação(*lambda_*): 0, 0.5, 1

A seguir, gráficos demonstrando o efeito da correlação entre mudanças:

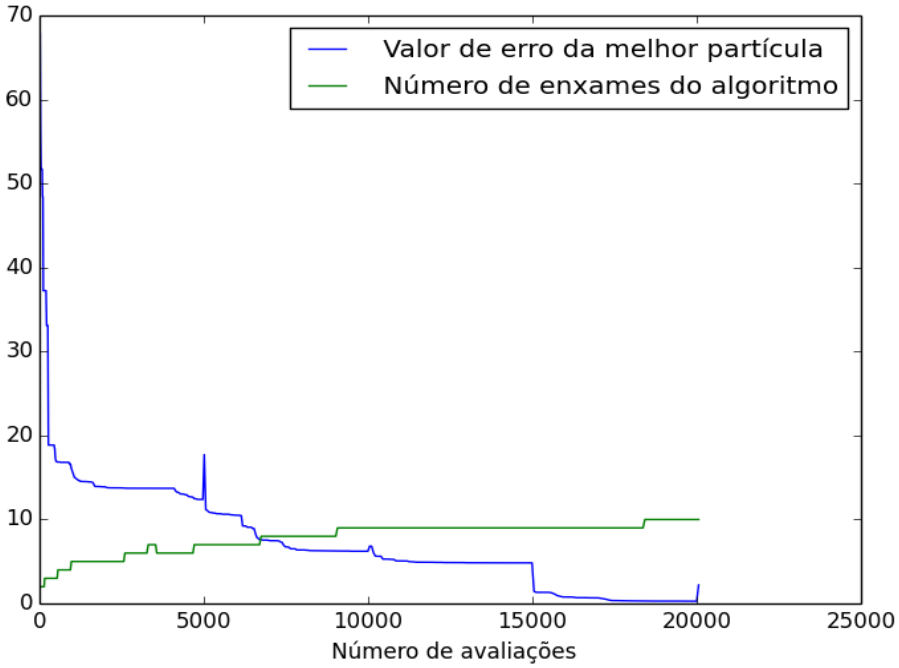

Figura 5.9: *Lambda_* = 0

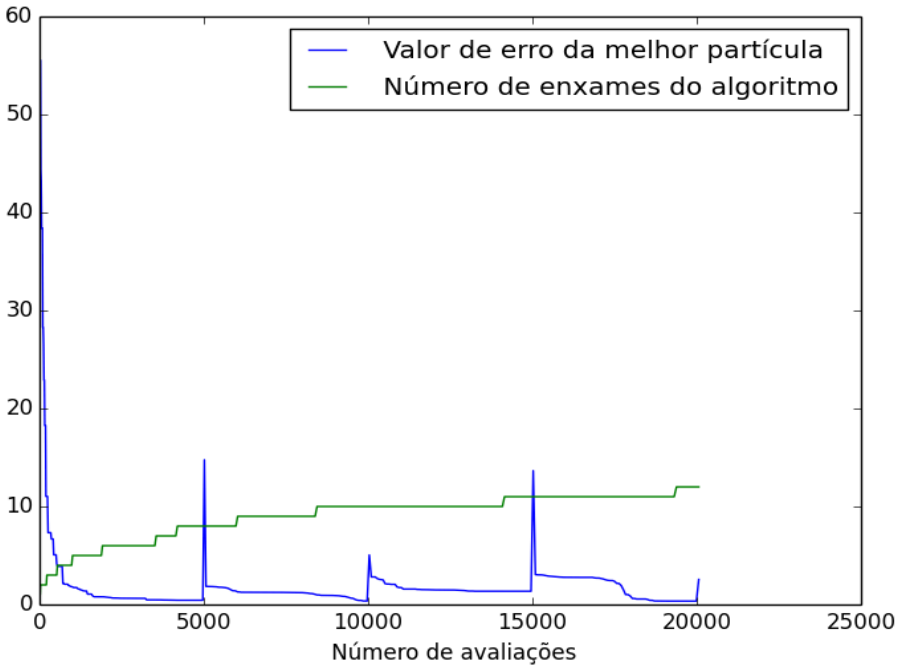

Figura 5.10: *Lambda_* = 0.5

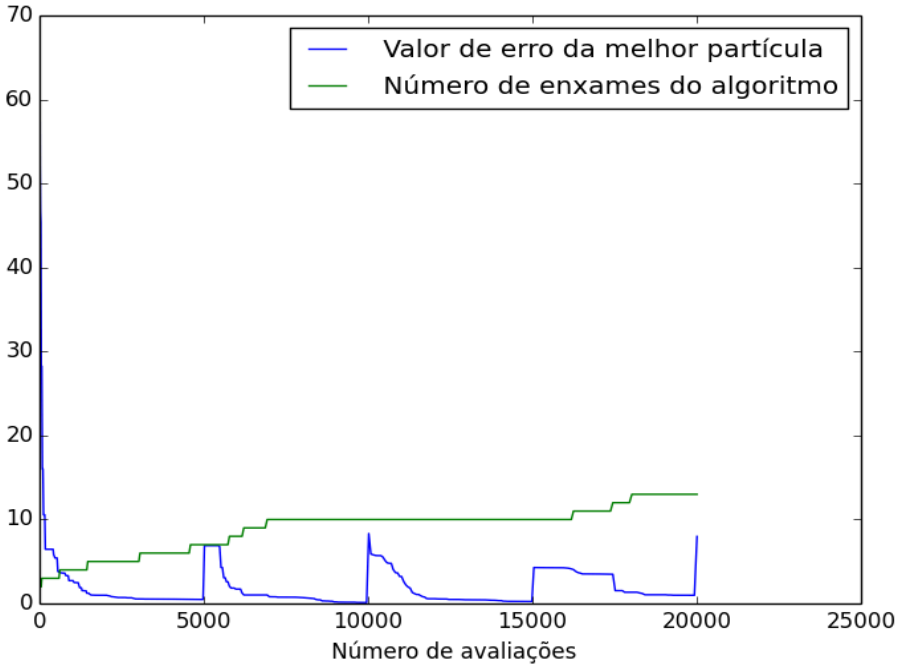

Figura 5.11: $Lambda_ = 1$

Como esperado, na Fig. 5.9 mostra que nenhuma correlação entre mudanças afeta a convergência do algoritmo e, de acordo como o cenário que fica, a convergência pode ser mais lenta. A medida que o número de enxames aumenta, mudanças inesperadas são melhor cobertas e o desempenho melhora um pouco.

Já nas Fig. 5.10 e Fig. 5.11 , como houve certa correlação, a convergência ficou mais fácil tanto com 50

5.2.5 Número de dimensões no espaço de busca

O número de dimensões no espaço de busca refere-se ao parâmetro NDIM(“ número de dimensões “) do algoritmo Multi-PSO. Esse parâmetro é o número de dimensões que uma partícula tem em seu espaço.

Foram feitos testes no cenário 2 com número de dimensões: 5 e 15. A seguir, gráficos de desempenho do algoritmo para cada dimensão:

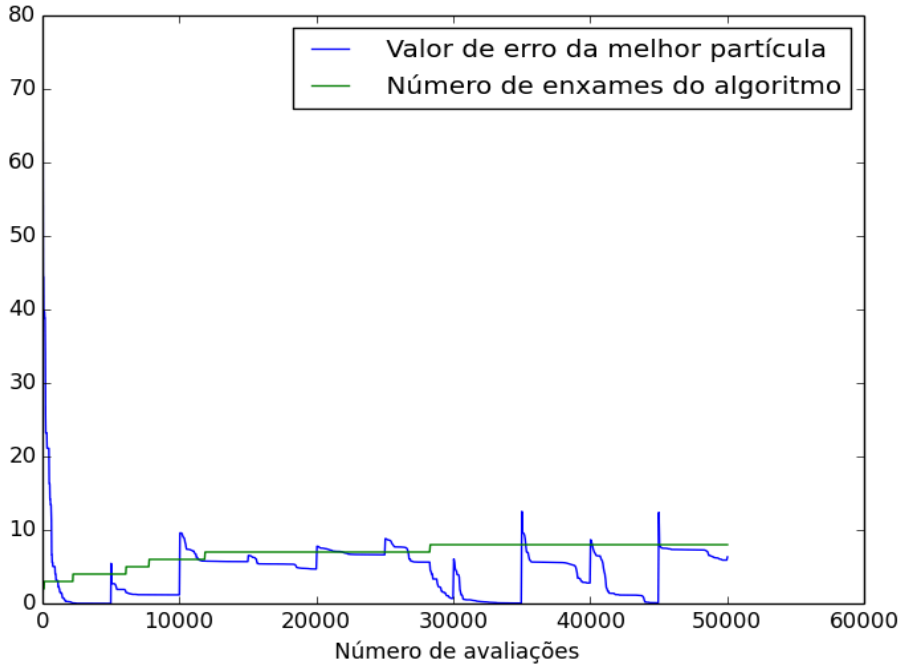

Figura 5.12: 5 dimensões

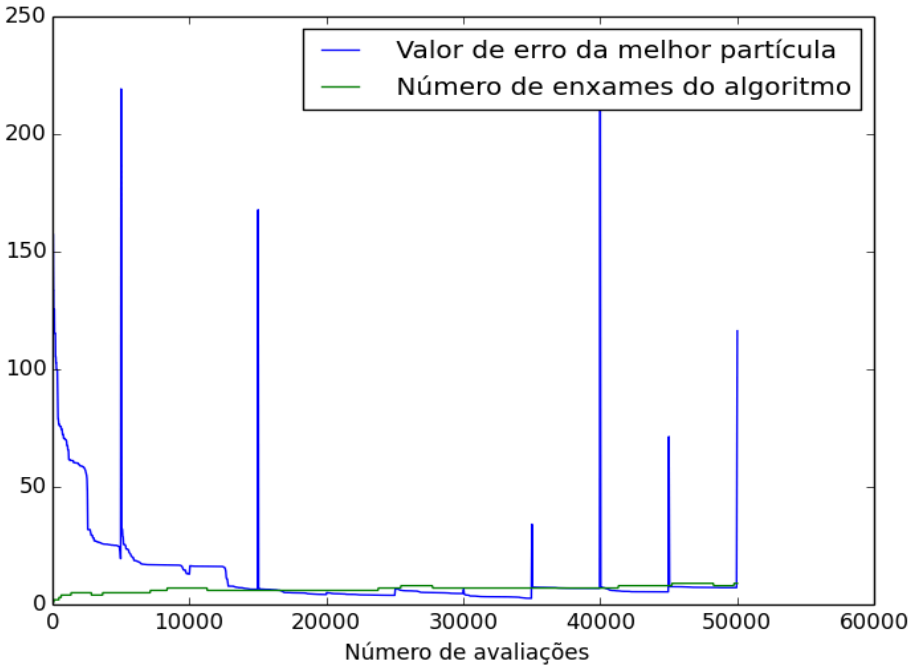

Figura 5.13: 15 dimensões

Como esperado, um problema com 15 dimensões é muito mais complexo que um problema com 5. Na Fig. 5.12, o erro da melhor partícula fica no máximo em 10. Ja, na Fig. 5.13,

dificilmente algum enxame consegue convergir para um ótimo global e o erro chega a mais de 200 em 5000 e 40000 avaliações por exemplo e fica entre 5 e 25 no geral. É um problema muito mais difícil para o algoritmo Multi-PSO. Também seria uma parte importante para otimização do algoritmo. Uma idéia seria o uso de “threads” para multi processamento.

O tempo de execução do algoritmo para 5 dimensões foi de 4.901 segundos e para 15 dimensões foi de 7.327 segundos. Uma diferença muito grande considerando que o cenário 2 do MPB é um problema relativamente fácil.

5.3 Características do algoritmo

A seguir, 2 características do algoritmo foram analisadas quanto a seu desempenho: a anti-convergência e o numero de dimensões no espaço de busca de cada partícula.

5.3.1 A Anti-convergência

O mecanismo de convergência proposto em [Blackwell,] tem como base 2 coisas: adicionar um enxame quando todos os enxames convergirem e remover 1 enxame quando tem muitos sem convergir, n>NEXCESS. O pior enxame é removido quando o número de enxames livres, ou seja, sem convergir, é maior que NEXCESS.

A remoção de enxames é necessária porque geralmente enxames livres que ficam vagando sem convergir podem afetar o desempenho ao consumir algumas avaliações de função.

Blackwell[Blackwell,] fez testes para verificar o impacto de NEXCESS no algoritmo e a achar um número que pudesse não afetar muito o desempenho. Os parâmetros utilizados foram: npeaks igual a 10 e 200 picos e 500.000 avaliações sem período de mudança dentro da execução. A seguir, uma tabela com resultados:

Tabela 5.2: A tabela mostra a execução do algoritmo para o número de picos p=10 e p=200. Os números são a média dos erros da melhor partícula. Dados retirados de [Blackwell,]

NEXCESS	**npeaks = 10** **Offline-error do MPB**	**npeaks = 150** **Offline-error do MPB**
1	1.9729028458116666	2.5383187958098343
2	1.879641879674056	2.398361911062971
3	1.7699076299648027	2.386396596554201
4	1.8033988974332964	2.372590853208213
5	1.8013758537004643	2.365026825401844
6	1.8010120393728533	2.3651325663361167
7	1.8010120393728533	2.3651325663361167
infinito	1.8010120393728533	2.3651325663361167

Esse teste mostra que para NEXCESS=3 ou mais, o erro tende a ficar constante, ou seja, o melhor NEXCESS seria 3 para esse caso. A conclusão de Blackwell foi que talvez o algoritmo poderia, durante a execução, achar um valor bom para NEXCESS e usá-lo. Para isso, um critério de convergência mais sofisticado poderia ser criado. Por exemplo, o critério de convergência poderia levar em conta tanto o diâmetro do enxame quanto a taxa de melhora de pg, que seria a melhor posição com melhor fitness de todos os enxames[Blackwell,].

5.3.2 Número de partículas

O número de partículas refere-se ao parâmetro NPARTICLES do algoritmo que é a quantidade de partículas que cada enxame tem. Foram feitos testes para avaliar se essa quantidade afeta o desempenho do algoritmo.

Os testes foram feitos no cenário 2 do MPB com 5 partículas e com 15 partículas e também, 50 picos.

A seguir, gráficos de desempenho do algoritmo para cada quantidade de partículas:

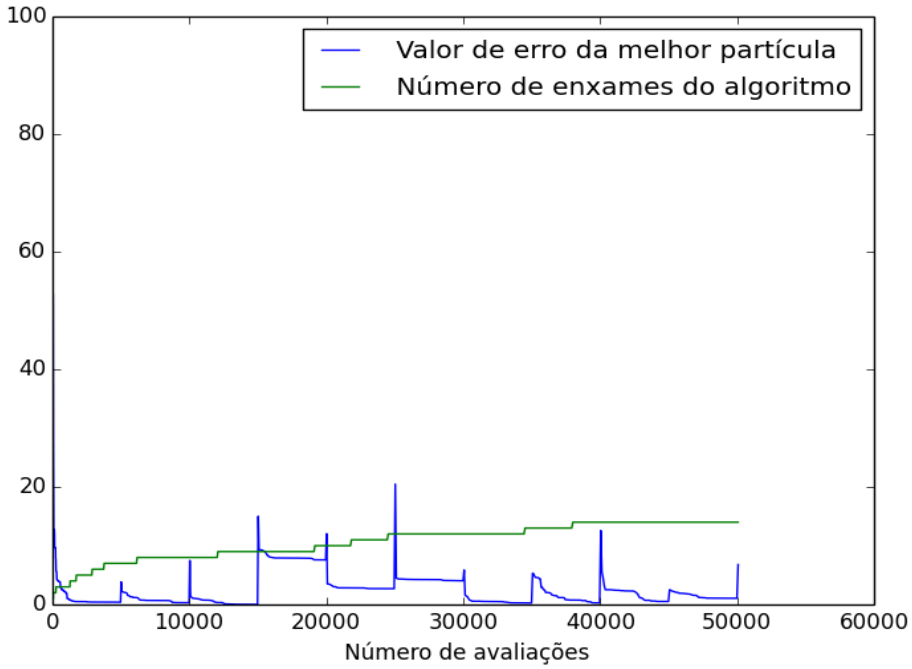

Figura 5.14: 5 partículas

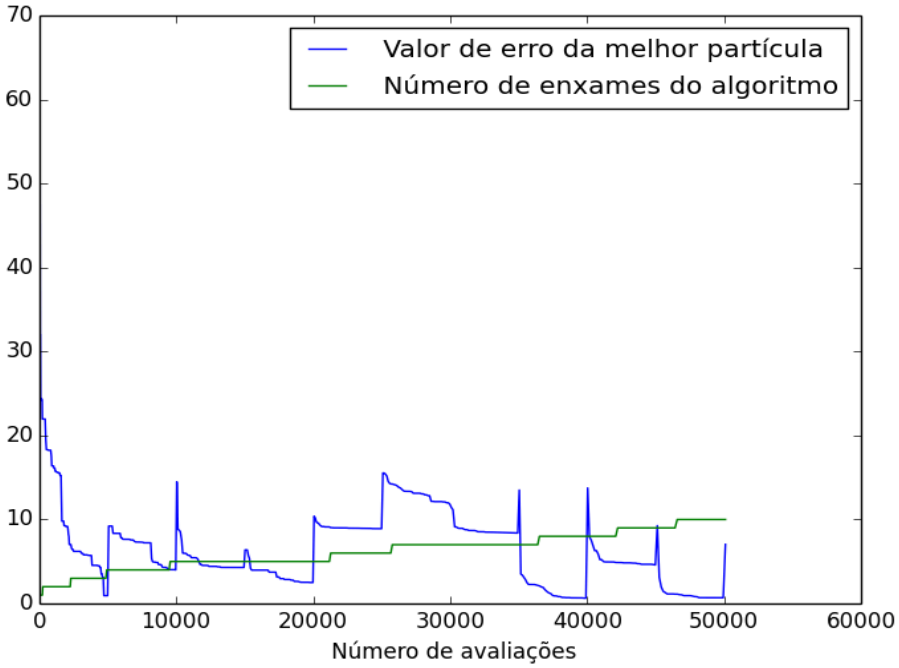

Figura 5.15: 15 partículas

Fica evidente pela comparação das figuras Fig. 5.14 e Fig. 5.15 que o algoritmo tem mais dinamismo e melhor resultado com 5 partículas. Isso ocorre porque 15 partículas podem dificultar os mecanismos do algoritmo de movimentação e manutenção da diversidade por ter que fazer muito mais avaliações e comparações, bem como partículas podem convergir de maneira confusa em algumas ocasiões por haver mais possibilidades de melhores partículas.

Capítulo 6

Conclusões e Trabalhos Futuros

Esse trabalho apresentou o algoritmo canônico PSO, que é um algoritmo de otimização baseado em enxame de partículas. Esse algoritmo deve ser modificado para ter um desempenho boa em ambientes do mundo real, simulados pelo MPB. Pesquisadores consideram a abordagem de multi-populações como a melhor abordagem para problemas do MPB. Então, o algoritmo Multi-PSO foi apresentado, que usa a analogia de átomo em cada enxame e técnicas para manter a diversidade, que são: anti-convergência, exclusão. A anti-convergência foca em manter pelo menos 1 enxame livre para percorrer o espaço de busca e a exclusão faz com que apenas 1 enxame possa convergir em cada pico ou ótimo. Consiste de uma função objetivo que muda com o tempo.

Após isso, foram feitos alguns experimentos para verificar a eficácia de certos parâmetros do MPB e do algoritmo de Multi-Enxame. O número de picos interfere no momento de mudança no ambiente, pois quantos mais picos existem, mais mudanças ocorrem, o que impacta no desempenho. Um parâmetro interessante verificado foi a severidade da mudança, e constatou-se que as mudanças impactam muito no desempenho algoritmo, principalmente as mais severas. Seria uma área para pensar melhoras no algoritmo futuramente. O período de mudança é muito importante, pois problemas de otimização dinâmica são baseados nos momentos de mudança e quando ocorrem. Ambientes caóticos, onde ocorrem muitas mudanças, dificultam a convergência para o ótimo global no algoritmo Multi-PSO. Essa seria uma outra área e bastante importante para pensar em melhoras no algoritmo. No mundo real, mudanças sucessivas podem ter correlação em diversos níveis. A medida que mais enxames são criados, pouca correlação tem um impacto reduzido no desempenho. Todos esses parâmetros analisados são de ambiente e não podem ser controlados no mundo real.

Um outro parâmetro verificado é o número de dimensões no espaço de busca. Certos problemas podem requerer um número maior de dimensões para serem resolvidos. O algoritmo Multi-PSO tem muita dificuldade de convergir quando a dimensionalidade é maior. Esse impacto grande no desempenho é por causa do aumento em unidades de grandeza da complexidade do problema. Seria uma ótima área a ser estudada futuramente.

Dois parâmetros especiais do algoritmo foram testados: a anti-convergência e o número de partículas. Concluiu-se que o número de enxames livres para manter a diversidade tem um ótimo para cada situação e que seria interessante ter um mecanismo mais elaborado de anti-convergência para que seja possível achar um número de enxames livres satisfatório, ou NEXCESS no algoritmo , em tempo de execução. As movimentos das partículas são a base do algoritmo PSO e Multi-PSO também. Enxames com 5 partículas têm um dinamismo maior e um funcionamento mais rápido que 15 partículas. Esses parâmetros são especiais do algoritmo e com certeza seriam bons alvos de estudo.

Como problemas no mundo real são dinâmicos e variados, um trabalho futuro seria em produzir uma variante mais auto-adaptável do algoritmo Multi-PSO, ou seja, um algoritmo que possa, em tempo de execução, otimizar sozinho a maior quantidade de parâmetros sem isso precisar ser feito á mão. Por exemplo a re-diversificação.

Finalmente, um outro trabalho futuro seria aplicar o algoritmo de multi-enxames em problemas reais e fazer a mesma verificação da eficácia dos parâmetros, para assim moldar e até mesmo atribuir mudanças ao algoritmo que favoreçam a resolução satisfatória desses problemas.

Referências Bibliográficas

[Blackwell,] Blackwell, T. Particle swarm optimization in dynamic environments. Dissertação de Mestrado, Goldsmiths College London, London.

[Blackwell, 2003a] Blackwell, T. (2003a). Swarm music: Improvised music with multi-swarms. *ProcAISB '03 Symposium on artificial intelligence and creativity in arts and science*, página 41–49.

[Blackwell, 2001] Blackwell, T. M. (2001). Evolutionary approaches to dynamic environments - updated survey. *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, página 27–30.

[Blackwell, 2003b] Blackwell, T. M. (2003b). Particle swarms and population diversity i: Analysis. *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, página 9–13.

[Blackwell, 2003c] Blackwell, T. M. (2003c). Particle swarms and population diversity ii: Experiments. *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, página 14–18.

[Blackwell e Bentley, 2002] Blackwell, T. M. e Bentley, P. (2002). Don't push me! collision avoiding swarms. *In Congress on Evolutionary Computation*, página 1691–1696.

[Blackwell e Branke,] Blackwell, T. M. e Branke, J. Multi-swarms, exclusion and anti-convergence in dynamic environments. *IEEE transactions on Evolutionary Computation*.

[Blackwell e Branke, 2004] Blackwell, T. M. e Branke, J. (2004). Multi-swarm optimization in dynamic environments. *Applications of Evolutionary Computing*, 3005 of LNCS:489–500.

[Branke,] Branke, J. Implementação do moving peaks benchmark da biblioteca deap. `deap.gel.ulaval.ca/doc/default/api/benchmarks.html#deap.benchmarks.movingpeaks.MovingPeaks`. Acessado em 15/12/2016.

[Clerc e Kennedy, 2000] Clerc, M. e Kennedy, J. (2000). The particle swarm: explosion, stability and conver- gence in a multi-dimensional space. *IEEE transactions on Evolutionary Computation*, (6).

[C.Reynolds, 1987] C.Reynolds (1987). Herds and schools: a distributed behavioral model. *Computer Graphics*, (21):25–34.

[du Plessis e Engelbrech, 2013] du Plessis, M. e Engelbrech, A. (2013). Self-adaptive differential evolution for dynamic environments with fluctuating numbers of optima. Dissertação de Mestrado, Department of Computer Science - University of Pretoria - Pretoria, South Africa.

[Goossens et al., 1990] Goossens, M., Mittelbach, F. e Samarin, A. (1990). *Linear programming and network flows (2nd ed.)*. H. D., Wiley - NY.

[Hu e Eberhart, 2002] Hu, X. e Eberhart, R. (2002). Adaptive particle swarm optimisation: detection and response to dynamic systems. *In Proc Congress on Evolutionary Computation*, página 1666–1670.

[Iskandar, 1983] Iskandar, J. (1983). *The Fractal Geometry of Nature*. W. H. Freeman and Company.

[R.Eberhart e Y.Shi, 2001] R.Eberhart e Y.Shi (2001). *Swarm intelligence*. Morgan Kaufmann.