UNIVERSIDADE FEDERAL DO MARANHAO - UFMA
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET
CURSO DE CIENCIA DA COMPUTAÇÃO

GABRIEL BARROSO DE ARAUJO

CAMPEONATO MUNDIAL DE META-HEURÍSTICAS EVOLUTIVAS

São Luís
2013

GABRIEL BARROSO DE ARAUJO

CAMPEONATO MUNDIAL DE META-HEURÍSTICAS EVOLUTIVAS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Alexandre César Muniz de Oliveira

São Luís
2013

Araújo, Gabriel Barroso de

Campeonato mundial de meta-heurísticas evolutivas/ Gabriel Barroso de Araújo. -- 2013.

98 f.

Orientador: Alexandre César Muniz de Oliveira.

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2013.

1. Algoritmos 2. CEC 2005 3. Heurística 4. Meta-heurística 5. Otimização contínua I. Título.

CDU 004.421.2

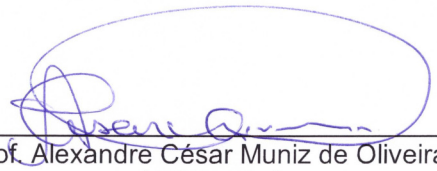
GABRIEL BARROSO DE ARAUJO

CAMPEONATO MUNDIAL DE META-HEURÍSTICAS EVOLUTIVAS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em: 04 / 11 / 2013

BANCA EXAMINADORA



Prof. Alexandre César Muniz de Oliveira
(Orientador)

Portela

Prof. Carlos Eduardo Portela Serra de Castro
(Examinador)

Mário Teixeira

Prof. Mário Antonio Meireles Teixeira
(Examinador)

À minha querida avó paterna Henilde Medeiros
e à minha noiva Lorenna Cabral, pelo amor
imensurável, pelos cuidados, dedicação e pelo
constante incentivo.

AGRADECIMENTOS

Agradeço primeiramente a Deus por permitir que tudo aconteça.

Agradeço ainda, à minha noiva Lorenna que sempre me apoiou em tudo o que fiz e à minha avó Henilde que sempre esteve ao meu lado desde que nasci até hoje me ajudando em todos os momentos, e aos meus pais pela educação, caráter e incentivo aos estudos que me deram.

Agradeço também ao meu orientador Alexandre César Muniz de Oliveira, pela dedicação e esforços dedicados a este trabalho.

“Aqueles que param esperando as coisas melhorarem acabarão descobrindo mais tarde que aqueles que não pararam estão tão na frente que não poderão ser mais alcançados."

Rui Barbosa

RESUMO

Estudo que investiga e tenta apontar qual é a melhor meta-heurística evolutiva existente na atualidade, para otimização de funções. Para realizar esta tarefa, foi necessário fazer um levantamento sobre otimização, meta-heurísticas e algum tipo de competição que permitisse comparar essas meta-heurísticas de forma justa e coerente. Um evento que fez um comparativo com essas características foi o CEC 2005. Então este estudo faz um levantamento detalhado acerca dos algoritmos (meta-heurísticas) que competiram, da competição em si e de suas regras, as funções utilizadas para se avaliar os algoritmos e ao final, tem-se uma análise detalhada sobre o resultado dessa competição. Foi constatado que a melhor meta-heurística de otimização existente até a data da elaboração deste trabalho é o IPOP-CMA-ES, também conhecido por G-CMA-ES. Porém, é preciso ser esclarecido que esse foi o resultado constatado baseando-se nos resultados do evento, de modo que não obrigatoriamente esse seja o melhor algoritmo, já que ele não foi o mais rápido e sua implementação é razoavelmente complicada. Então, o importante é analisar bem a situação e decidir qual é o algoritmo mais adequado a ser aplicado a ela.

Palavras-chave: Algoritmos; CEC 2005; Heurística; Meta-heurística; Otimização Contínua.

ABSTRACT

Study that investigates and tries to point out which is the best existing evolutionary metaheuristic today, for functions optimization. To accomplish this task, it was necessary to make a survey on optimization, metaheuristics and some sort of competition which would allow to compare these metaheuristics in a correct and consistent way. An event that made a comparison with these characteristics was the CEC 2005. So this study is a detailed survey on the algorithms (metaheuristics) that competed, the contest itself and its rules, the functions used to evaluate the algorithms, and at the end, there is a detailed analysis on the outcome of this competition. It was found that the best optimization metaheuristic existing up to the date of preparation of this work is the IPOP-CMA-ES, also known as G-CMA-ES. But we must be clear that this was the result found based on the results of the event, so this is not necessarily the best algorithm, since it was not the fastest one and its implementation is fairly complicated. So, it is important to analyze the situation well and decide which is the best algorithm to be applied to it.

Key Words: Algorithms; CEC 2005; Heuristic; Metaheuristic; Continuous Otimização Contínua

LISTA DE ILUSTRAÇÕES

Figura 01 – Métodos de Otimização Clássicos 18
Figura 02 – Genealogia das Meta-Heurísticas 20
Quadro 01 – Procedimentos do BLX-GL50 23
Quadro 02 – Aplicação do LS 26
Quadro 03 – Algoritmo CoEVO 29
Quadro 04 – Algoritmo DE 30
Figura 03 – Comportamento do DE 31
Figura 04 – DMS-L-PSO Search 34
Quadro 05 – Pseudocódigo do DMS-L-PSO 35
Quadro 06 – Pseudocódigo do $\mathrm{EDA}mvg$ 36
Figura 05 – Comportamento do EDA 37
Quadro 07 – Pseudocódigo do CMA-ES 40
Figura 06 – Comportamento do CMA-ES 41
Figura 07 – Prole gerada por um operador PNX 54
Quadro 08 – Valores de *N* adotados por SPC-PNX para cada função 55
Figura 08 – Mapa 3-D para função F1 2-D 57
Figura 09 – Mapa 3-D para função F2 2-D 58
Figura 10 – Mapa 3-D para função F3 2-D 59
Figura 11 – Mapa 3-D para função F4 2-D 60
Figura 12 – Mapa 3-D para função F5 2-D 61
Figura 13 – Mapa 3-D para função F6 2-D 63
Figura 14 – Mapa 3-D para função F7 2-D 64
Figura 15 – Mapa 3-D para função F8 2-D 65
Figura 16 – Mapa 3-D para função F9 2-D 66
Figura 17 – Mapa 3-D para função F10 2-D 67
Figura 18 – Mapa 3-D para função F11 2-D 68
Figura 19 – Mapa 3-D para função F12 2-D 69
Figura 20 – Mapa 3-D para função F13 2-D 70
Figura 21 – Mapa 3-D para função F14 2-D 71
Figura 22 – Mapa 3-D para função F15 2-D 72
Figura 23 – Mapa 3-D para função F16 2-D 73
Figura 24 – Mapa 3-D para função F17 2-D 74
Figura 25 – Mapa 3-D para função F18 2-D 75
Figura 26 – Mapa 3-D para função F19 2-D 76
Figura 27 – Mapa 3-D para função F20 2-D 77
Figura 28 – Mapa 3-D para função F21 2-D 78
Figura 29 – Mapa 3-D para função F22 2-D 79
Figura 30 – Mapa 3-D para função F23 2-D 80
Figura 31 – Mapa 3-D para função F24 2-D 82
Figura 32 – Gráfico do Resultado Geral 85
Figura 33 – Gráfico do Resultado para funções unimodais 86

Figura 34 – Gráfico do Resultado para funções multimodais 88
Quadro 10 – Ranking dos algoritmos para funções multimodais resolvidas 88
Quadro 11 – Ranking dos algoritmos para funções multimodais não resolvidas 89
Quadro 12 – Ranking Geral das meta-heurísticas de otimização do CEC 2005 90
Quadro 13 – Classificação Final.....91

LISTA DE SIGLAS

BLX-GL50	Blended Crossover García-Martínez Lozano 50
BLX-MA	Blended Crossover Memetic Algorithm
CEC	Congress on Evolutionary Computation
CoEVO	CoEvolution
DE	Differential Evolution
DMS-L-PSO	Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search
EDA	Estimation of Distribution Algorithm
IEEE	Institute of Electrical and Electronics Engineers
IPOP-CMA-ES	Increasing Population Covariance Matrix Evolutionary Strategy
K-PCX	K PCX Crossover
LR-CMA-ES	Restart Covariance Matrix Evolutionary Strategy with Local Search
L-SaDE	Self-adaptive Differential Evolution with Local Search
SPX-PNX	Scaled Probabilistic Crowding with PNX Crossover

SUMÁRIO

1 INTRODUÇÃO 12

2 META-HEURÍSTICAS 14

2.1 OTIMIZAÇÃO 14
2.2 MÉTODOS DE OTIMIZAÇÃO 15
2.3 META-HEURÍSTICAS DE OTIMIZAÇÃO 18

3 COMPETIDORES 21

3.1 BLX-GL50 21
3.2 BLX-MA 24
3.3 CoEVO 27
3.4 DE 29
3.5 DMS-L-PSO 32
3.6 EDA 35
3.7 IPOP-CMA-ES (ou G-CMA-ES) 39
3.8 K-PCX 43
3.9 LR-CMAES (ou L-CMA-ES) 46
3.10 L-SaDE 48
3.11 SPC-PNX 52

4 COMPETIÇÃO 56

4.1 METODOLOGIA, FUNÇÕES E MÉTRICAS 57
4.2 RESULTADOS, GRÁFICOS E RANKING 83
4.3 CONSIDERAÇÕES FINAIS 91

5 CONCLUSÃO 95

REFERENCIAS 96

1 INTRODUÇÃO

Otimização é a ação de se encontrar o melhor resultado de uma função, ou seja, encontrar seu ponto ótimo. Pela definição, parece simples, mas não é, dependendo da função abordada. Para isso foram criados os métodos de otimização, onde inicialmente têm-se os métodos exatos (ou naturais), que buscavam o valor exato da função, porém eram muito dependentes desta.

Para corrigir esta e mais outras deficiências dos métodos naturais, foram criados os métodos aproximados, que como o nome já diz, não buscam o resultado exato, mas um resultado aproximado, que seja muito bom, aceitável. Dentre esses métodos destacam-se os métodos heurísticos (ou simplesmente heurísticas), que dependiam muito menos da função matemática do que os métodos naturais. Desta forma, era possível aplicar a mesma heurística em diferentes funções matemáticas.

Algum tempo depois, com o intuito de melhorar ainda mais as heurísticas, foram desenvolvidas as meta-heurísticas, que tornaram as heurísticas ainda mais independentes da função matemática. As meta-heurísticas são guias para as heurísticas, de modo que as torna mais genéricas e aplicáveis a um maior número de funções, com o mínimo de adaptações (ou nenhuma adaptação) entre suas aplicações em diversas funções matemáticas.

Tendo o tema otimização em foco, uma divisão do *Institute of Electrical and Electronics Engineers* (IEEE), resolveu criar uma conferência, em parceria com a *Evolutionary Programming Society*, para debater sobre esse assunto. Essa conferência é o *Congress on Evolutionary Computation* (CEC), que tem a finalidade de reunir pessoas com interesse e trabalhos relacionados à Computação Evolucionária, bem como lançar desafios em busca de novidades e melhorias na área.

O trabalho em questão enfatiza a otimização contínua e utilizou como base de estudo o CEC 2005, uma vez que foi a maior conferência já ocorrida relacionada a esse assunto, e nela se obtiveram os melhores resultados, bem como os mais detalhados. No evento foram recebidos 17 *papers* referentes aos algoritmos competidores, sendo que apenas 11 foram aceitos. São eles: BLX-GL50, BLX-MA, Algoritmo CoEVO, Evolução Diferencial, DMS-L-PSO, EDA, IPOP-CMAES, K-PCX, LR-CMAES, L-SaDE e SPC-PNX. Alguns desses algoritmos são bem parecidos entre si, diferindo apenas em pequenos detalhes. Outros são versões modificadas

de meta-heurísticas e há ainda versões híbridas de meta-heurísticas, etc. São apresentados seus resultados de desempenho e suas características, tentando fazer um comparativo de desempenho entre eles.

O objetivo geral deste trabalho é apresentar os melhores algoritmos de otimização contínua que competiram no CEC 2005 (*Congresso of Evolutionary Computation, 2005*), bem como a competição em si, metodologia de comparação, *benchmark* empregado, além de aspectos relevantes para o entendimento de otimização e algoritmos heurísticos.

Os objetivos específicos foram, portanto, realizar uma pesquisa, um levantamento de dados acerca dos algoritmos, da competição, bem como dos resultados, e demonstrar ao final, um avaliação acerca dessa disputa e tentar eleger a melhor meta-heurística existente na atualidade.

Por fim, vale ressaltar que, através deste, pretende-se documentar os algoritmos competidores, referenciando suas origens, o *benchmark* e as metodologias utilizados no CEC 2005, que tem se tornado referência na área de otimização contínua, sem, todavia, aprofundar-se nas explicações sobre cada algoritmo nem nos métodos de otimização pioneiros. O objetivo deste trabalho não está relacionado aos algoritmos pioneiros que deram origem a esses competidores, e supõe-se que o leitor já tenha um conhecimento prévio sobre estes métodos e heurísticas pioneiros, e assim, o presente trabalho não possui nenhuma explanação referente a algoritmos genéticos, busca local, estratégias evolutivas, busca tabu, método do gradiente, e outros métodos de otimização e heurísticas que não sejam os 11 algoritmos competidores.

O trabalho divide-se em cinco partes. Na primeira delas tem-se essa introdução; O segundo item aborda os conceitos e caraterísticas relacionados a meta-heurísticas; No terceiro item são apresentados cada uma das mata-heurísticas competidoras que fizeram parte do CEC 2005, explicação de seus funcionamentos e seus algoritmos; No quarto item apresenta-se a competição, onde tem-se a metodologia e a análise dos métodos utilizados, funções, gráficos e rank e o comparativo entre os competidores; E ao final apresenta-se a conclusão acerca do trabalho.

2 META-HEURÍSTICAS

2.1 OTIMIZAÇÃO

Otimização significa, em Matemática, o estudo de problemas com o objetivo de minimizar ou maximizar o resultado de uma função, por meio da escolha sistemática de valores de variáveis em um conjunto viável. Essas variáveis podem assumir valores inteiros, caracterizando um Domínio Discreto, ou podem assumir valores reais, caracterizando um Domínio Real. Otimização Contínua é o tipo de Otimização em que se tem um problema com Domínio Real.

Bennaton (2013, não paginado) em seus estudos afirma que:

> [...] Otimização acabou se constituindo numa vasta e sólida área do conhecimento. Uma área híbrida, eclética e pragmática. Com um pé na matemática e outro pé na computação. Que se nutre das ciências exatas, das biológicas, das tecnológicas. Que se dedica a solucionar problemas independentemente do contexto onde surgem. Problemas práticos e efetivos. Problemas relativos a como determinar uma alternativa melhor que outra, dentro de um universo dado. Otimização liga-se à matemática através da investigação dos máximos e mínimos locais de funções. Isto é, dos "valores estacionários da função" (que ocorrem quando a derivada se anula) se quisermos utilizar a linguagem apropriada.

Ainda utilizando os estudos de Bennaton (2013, não paginado):

> [...] Na matemática é costume se concentrar na existência de máximos e mínimos locais. Condições necessárias, condições suficientes. Na otimização não basta isto. [...] Tem que se dizer como determinar, efetivamente, os extremos procurados. Na verdade, este "como" acaba se impondo. Na otimização, a metodologia de cálculo é que acaba virando o prato principal. Acontece que nem sempre o "ótimo" sai de pronto, num único cálculo. É necessário um processo sistemático de busca. Um procedimento iterativo de cálculos que, passo a passo, de candidato a candidato, permita-nos ir melhorando a seleção. Até que o ótimo seja encontrado, ou até que estejamos satisfeitos.

Um problema de otimização pode ser representado da seguinte forma:

Dados: uma função $f: A \rightarrow \mathbb{R}$, de algum conjunto A de números Reais;

Buscando: um elemento x_o em A tal que $f(x_o) \leq f(x)$ para todo x em A se for de minimização, ou $f(x_o) \geq f(x)$ para todo x em A caso seja de maximização.

Wikipedia (2013a, não paginado) completa ainda que:

> Tal formulação é chamada de um problema de otimização ou um problema de programação matemática (um termo não diretamente relacionado à programação de computadores, mas ainda em uso, por exemplo, na programação linear). Muitos problemas do mundo real e teóricos podem ser modelados nessa estrutura geral. Problemas formulados usando esta técnica nos campos da física e da visão computacional podem se referir à técnica como minimização de energia, tratando o valor da função f como representativo da energia do sistema sendo modelado.

Geralmente, A é algum subconjunto do espaço euclidiano $\mathbb{R}^n$, que é normalmente definido por um conjunto de restrições, igualdades ou desigualdades que os membros de A devem satisfazer. Este domínio A de f, é o que se chama de *espaço de busca* ou *conjunto de escolha*, ao mesmo tempo em que os elementos de A são chamados de *soluções candidatas* ou *soluções viáveis*.

Alternadamente, a função f é chamada de *função-objetivo*, *função de custo* (minimização), *função utilidade* (maximização), ou, em determinados campos, *função de energia*, ou *energia funcional*. *Solução ótima* é a definição de uma solução viável que minimiza (ou maximiza, se esta é a intenção) a função objetivo.

Nesse contexto, a principal ideia é encontrar, dentre um conjunto de variáveis, a mais perfeita combinação de valores para essas variáveis, com o intuito de conseguir alcançar um determinado objetivo. Diante dos fatos que os problemas tornam-se cada vez mais complexos, a evolução cada vez maior da tecnologia, bem como a sofisticação dos recursos computacionais, acabam motivando a busca e o avanço por melhores técnicas de otimização.

2.2 MÉTODOS DE OTIMIZAÇÃO

Os primeiros métodos desenvolvidos para se otimizar uma função ou problema foram chamados de Métodos Exatos ou Métodos Naturais. Esses métodos buscam uma solução exata, perfeita, definitiva, para a função dada. Um dos primeiros métodos desse tipo foi o método da descida. Este método e consiste em gerar uma sequencia de iterandos $\{x^k\}$, tal que a sequencia $\{f(x^k)\}$ é decrescente. Cada iteração para se encontrar o próximo valor de $f(x)$ é denominada de passo. Deve-se notar que como a sequencia é decrescente, esse método é designado a resolver problemas de minimização.

Uma das primeiras ideias para se implementar o método da descida foi o método do gradiente, ou método baseado no gradiente. Este método é simples e seu conceito é devido a Cauchy. Como foi dito anteriormente, ele é uma implementação do método da descida, porém faz uso de um vetor chamado “gradiente”, que define a direção de descida e o tamanho do “passo” de cada iteração, inserindo robustez ao método da descida.

Este método tem um baixo custo computacional e funciona melhor com funções convexas, pois em funções côncavas, pode “ficar estagnado” facilmente em algum ótimo local em vez de continuar a busca pelo ótimo global. Além disso, possui taxa de convergência local arbitrariamente lenta para problemas mal escalonados, sendo muitas vezes considerado ineficiente e não confiável.

Para tentar solucionar as deficiências do método do gradiente, foram criados outros métodos baseados no método da descida, mas todos muito semelhantes ao método do gradiente, nos quais a principal mudança é o vetor que define o próximo passo. Sendo assim, são métodos mais robustos, mas possuem um custo computacional mais elevado, e dependendo do caso, o método do gradiente acaba sendo mais vantajoso, apesar de menos robusto.

Para contornar essa situação foram desenvolvidos os Métodos Aproximados, que são métodos aplicáveis a diversas situações/funções e que consomem menos recursos computacionais. Dentre esses métodos, destacam-se os Métodos Heurísticos, que são algoritmos exploratórios com o intuito de solucionar problemas, e possuem uma característica interessante: normalmente não implementam computacionalmente um conhecimento especializado. Uma boa explicação para tal definição é que, se um método heurístico fosse ser usado para solucionar uma equação de segundo grau, não seria necessário que fizesse uso da fórmula de Báscara, mas procuraria de alguma forma, uma solução que atendesse a essa equação. Devido a essa característica, esses algoritmos são classificados como “busca cega”, e são muitas vezes tratados mais como um Método de Busca do que como um Método de Otimização propriamente dito.

Fabrício Bueno (2009, p. 5) afirma que:

> Uma solução ótima de um problema nem sempre é o alvo dos métodos heurísticos, uma vez que, tendo como ponto de partida uma solução viável, baseiam-se em sucessivas aproximações direcionadas a um ponto ótimo. Logo, estes métodos costumam

> encontrar as melhores soluções possíveis para problemas, e não soluções exatas, perfeitas, definitivas.

O autor relata ainda em seus estudos, que essa falta de precisão ou subjetividade dos métodos heurísticos não é uma deficiência, mas uma particularidade análoga à inteligência humana, uma vez que no dia-a-dia as pessoas resolvem diversos problemas sem conhecê-los de forma precisa.

Os métodos heurísticos (ou simplesmente heurísticas) buscam encontrar uma solução boa, aceitável, para o problema dado, e não a solução ótima, exata, perfeita. Isto pode até ocorrer, por coincidência, mas não é o objetivo das heurísticas. Em seus estudos, Talbi (2009) descreve como métodos heurísticos mais populares os Algoritmos Genéticos, *Simulated Annealing*, Busca Tabu, GRASP, Colônia de Formigas e Colônia de Abelhas.

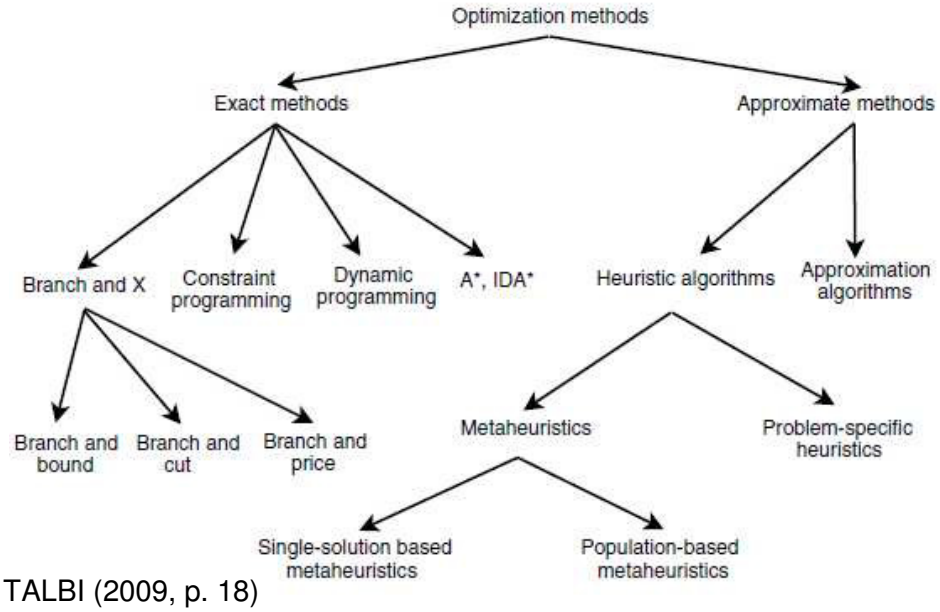
O objetivo principal das heurísticas é encontrar uma solução boa para vários problemas utilizando o mesmo algoritmo, enquanto que os métodos exatos necessitam que se tenha total controle sobre a função do problema, e a implementação do algoritmo é totalmente específica para aquele problema, não servindo para outra situação por menor que seja a diferença entre elas.

Miyazawa (2008) afirma que quando se tenta otimizar uma função ou um problema, dois itens devem ser levados em consideração:

1) Resolver o problema na otimalidade;
2) Resolver o problema em tempo computacional satisfatório.

Atualmente não existe um método/algoritmo que execute esses dois itens na sua otimalidade, sendo que ou se sacrifica o primeiro item para se obter um bom resultado no segundo, ou se sacrifica o segundo e privilegia o primeiro. Os Métodos Naturais privilegiam o primeiro item e sacrificam o segundo, enquanto que os Métodos Heurísticos dão prioridade ao segundo item e sacrificam o primeiro. Por essa divergência entre esses dois métodos, eles deram origem às duas vertentes no que diz respeito a métodos de otimização, como mostra a figura 01 a seguir:

Figura 01 – Métodos de Otimização Clássicos



Fonte: TALBI (2009, p. 18)

Tendo o tema otimização em foco, uma divisão do *Institute of Electrical and Electronics Engineers* (IEEE), a IEEE *Computional Intelligence Society*, em parceria com a *Evolutionary Programming Society*, resolveu criar uma conferência para debater sobre esse assunto. Essa conferência é o *Congress on Evolutionary Computation* (CEC), que teve seu inicio em 1999 e ocorre uma vez por ano, cada ano em uma cidade diferente. Esta conferência visa reunir pessoas com interesse e trabalhos relacionados à Computação Evolucionária, bem como lançar desafios em busca de novidades e melhorias na área.

2.3 META-HEURÍSTICAS DE OTIMIZAÇÃO

Meta-heurísticas são estratégias genéricas para a construção de heurísticas de otimização. Não há um consenso sobre seu conceito e muitas são as definições encontradas na literatura a respeito da mesma. Assim, optou-se por utilizar os estudos de Talbi (2009, p. 1) acerca da acepção de heurística:

> A palavra heurística tem sua origem da palavra do grego antigo *heuriskein*, que significa a arte da descoberta de novas estratégias (regras) para resolver problemas. O prefixo *meta*, também uma palavra grega, significa “nível superior”. O termo meta-heurística foi introduzido por F. Glover [...] Métodos de busca meta-heurísticos podem ser definidos como metodologias gerais de nível superior

> (modelos) que podem ser usados como estratégias de guia no desenvolvimento de heurísticas primárias para resolver problemas de otimização específicos.

Corroborando com a definição anterior, *Metaheuristics Network*[1] (2013, não paginado) acrescenta em seus estudos que:

> Uma meta-heurística é um conjunto de conceitos que pode ser utilizado para definir métodos heurísticos aplicáveis a um extenso conjunto de diferentes problemas. Em outras palavras, uma meta-heurística pode ser vista como uma estrutura algorítmica geral que pode ser aplicada a diferentes problemas de otimização com relativamente poucas modificações que possam adaptá-la a um problema específico.

Acerca das características gerais das meta-heuristicas, Becceneri (2013, não paginado) relata:

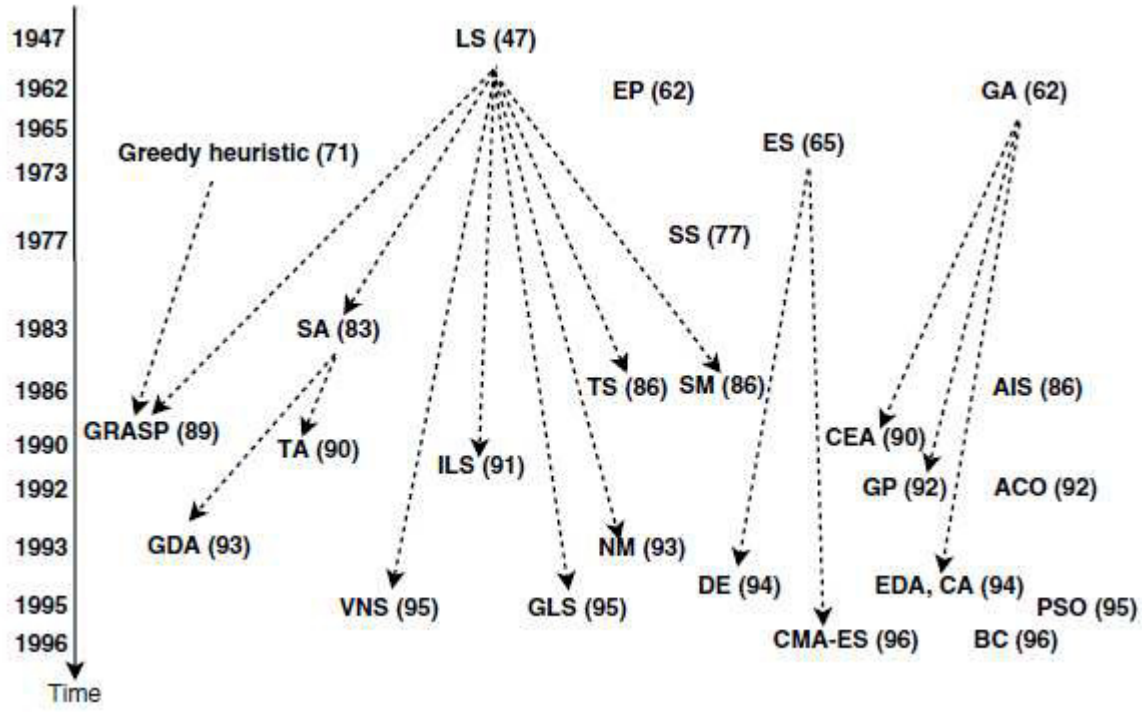
> São estratégias que guiam o processo de busca; O objetivo é explorar eficientemente o espaço de busca a fim de achar soluções ótimas ou quase-ótimas; São algoritmos variando de um simples procedimento de busca local a complexos processos de aprendizado; São algoritmos aproximados e usualmente não determinísticos; Podem incorporar mecanismos para evitar ficarem presas em áreas confinadas do espaço de busca; Não são específicas para um determinado problema; Podem usar um conhecimento específico do problema na forma de heurísticas que são controladas por uma estratégia de nível superior.

Dessa maneira, nota-se que as meta-heurísticas são muito semelhantes aos métodos heurísticos (ou heurísticas), sendo na verdade tratados como a mesma coisa pela maioria dos autores. O que se pode observar é que as meta-heurísticas são mais genéricas que os métodos heurísticos, e por isso podem ser aplicadas a um número maior de problemas, com um mínimo de adaptações de um para o outro. Isso não quer dizer que os métodos heurísticos são limitados a determinados problemas, porém, geralmente são formulados um para cada situação, sendo chamados, nesses casos, de *heurísticas especializadas*.

A figura 02 a seguir demonstra a genealogia das principais meta-heurísticas, incluindo as datas nas quais as mesmas foram desenvolvidas.

[1] *Metaheuristics Network web site*, 2013, não paginado. Tradução e adaptação do autor.

Figura 02 – Genealogia das Meta-Heurísticas



Fonte: TALBI (2009, p. 24)

Talbi (2009,apud Fingler 2013, p. 8) descreve essas meta-heurísticas da figura acima como: ACO (Otimização de colônia de formigas), AIS (Sistemas imunológicos artificiais), BC (Colônia de abelhas), CA (Algoritmos culturais), CEA (Algoritmos coevolucionários), CMA-ES (Estratégia de evolução adaptativa de covariância de matrizes), DE (Evolução diferencial), EDA (Algoritmos de estimativa de distribuição), EP (Programação evolucionária), ES (Estratégia de evolução), GA (Algoritmos genéticos), GDA (Algoritmos do grande dilúvio), GLS (Busca local guiada), GP (Programação genética), GRASP (Procedimento de busca gulosa adaptativa), ILS (Busca local iterativa), NM (Método de ruídos), PSO (Otimização de enxame de partículas), SA (Recozimento simulado), SM (Método de suavização), SS (Busca dispersa), TA (Aceitação de limiar), TS (Busca tabu) e VNS (Busca de vizinhança variável).

3 COMPETIDORES

Com o intuito de tentar determinar se existe uma meta-heurística melhor do que a(s) outra(s), e caso haja, qual é essa meta-heurística, o IEEE (*Institute of Electrical and Electronic Engineers*) resolveu realizar um evento internacional com o seguinte desafio: uma disputa entre as meta-heurísticas. Essa disputa foi uma espécie de campeonato, em que os competidores eram as meta-heurísticas e seus algoritmos iriam competir entre si, em busca dos melhores resultados, para se estabelecer um Ranking com as melhores e mais eficientes meta-heurísticas. Esse evento foi o CEC (*Congress on Evolutionary Computation*) que ocorreu em 2005, como mencionado anteriormente no trabalho.

Antes de 2005, o IEEE já havia realizado outras conferências com esse nome de CEC, pois esse congresso ocorre uma vez por ano desde o ano de 1999, sendo que em cada ano costuma ser abordado um tema diferente relacionado à computação evolutiva. Mas mesmo nos anos em que o tema era Otimização Contínua, ainda assim o CEC 2005 se destaca, pois foi o que apresentou os melhores e mais esclarecedores resultados até os dias atuais. Por esse mesmo motivo, ele tornou-se referência nesse assunto. A seguir, são apresentados os competidores, bem como a explicação de seus funcionamentos e seus algoritmos conforme a explanação e *papers* de diversos autores dispostos em um documento do IEEE.

3.1 BLX-GL50

De acordo com García-Martínez e Lozano (2013) o BLX-GL50 é um algoritmo genético híbrido, do tipo RCGA (*Real-coded Genetic Algorithm*), que faz uso de operadores de cruzamentos cêntricos dos pais, os PCCOs (*Parent-centric Crossover Operators*). Os PCCOs são uma família de operadores de parâmetro real que têm recebido uma atenção especial ultimamente. Em geral, esses operadores utilizam uma distribuição de probabilidade para criar uma descendência em um espaço de busca restrito em torno da região marcada por um dos pais, o progenitor feminino. A variedade dessa distribuição de probabilidade depende da distância entre o progenitor feminino e o outro progenitor envolvido no cruzamento, o progenitor masculino.

Ele é híbrido porque combina dois algoritmos genéticos de codificação real: um global RCGA e um local RCGA, sendo que esses dois algoritmos são de estado estacionário, baseados em processo FMD (*Female Male Differentiation*) e aplicam a estratégia de substituição do pior indivíduo. Além disso, o global RCGA usa o operador de cruzamento PBX-α, enquanto que o local RCGA usa o operador de cruzamento PCX.

O nome deste algoritmo é composto por uma mescla de siglas: o BLX é de *Blended Crossover*, enquanto que o GL vem das iniciais dos autores Garcia-Martínez e Lozano, e o número 50 é obtido de $P_G = 50\%$ (População obtida pelo algoritmo Global).

Partindo do princípio de que todos os cromossomos na população podem se tonar genitores masculino (*Male*) ou feminino (*Female*), é importante ressaltar que cada um dos dois têm diferentes papéis:

- Genitores femininos apontam para áreas de pesquisa/busca que receberão pontos de amostragem, enquanto que;
- Genitores masculinos são usados para determinar a extensão/alcance dessas áreas.

Sendo assim, é razoável dizer ainda que alguns cromossomos são bem adequados para atuarem tanto como genitores masculinos quanto como genitores femininos. Por isso, é de grande importância que se tenha um cuidado especial com a parte de diferenciação entre esses cromossomos, pois esta diferenciação é um dos fatores dos quais depende o sucesso deste algoritmo.

O código do BLX-GL50 é subdividido em 3 partes principais: a seleção dos cromossomos femininos pelo processo UFS (*Uniform Fertility Selection*), a seleção dos cromossomos masculinos através de uma variação do NAM (*Negative Assortative Mating*), e finalmente executa o algoritmo RCGA propriamente dito, que faz uso da estratégia de substituir o pior, RW (*Replace Worst*), que faz a substituição do pior indivíduo na população apenas se o novo indivíduo for melhor do que o atual. O quadro 01 a seguir demonstra esses três procedimentos:

Quadro 01 – Procedimentos do BLX-GL50

Pseudocódigo do UFS	1) $p_f \leftarrow$ Find the chromosome with less offsprings in G_F. 2) Increment the number of offsprings of p_f. 3) Return p_f.
Pseudocódigo do NAM	1) $L \leftarrow$ Set of n_{ass} randomly chosen chromosomes from G_M. 2) $p_M \leftarrow$ Find the chromosome in L that is the most dissimilar from the female parent p_f. 3) Return p_M.
Modelo de RCGA baseado em FMD	1) Initialize P with N randomly chromosomes. 2) Repeat until the stopCriterion is fulfilled. a. Select a female parent according to UFS from G_F. b. Select (μ - 1) male parents according to NAM from G_M. c. Apply the crossover operator to the parents in order to produce λ offspring. d. Insert the offspring in P by using the RW strategy.

Fonte: Adaptado de García-Martínez e Lozano (2013, p. 897)

Uma importante conclusão que pode ser observada é que o processo FMD permite desenvolver dois modelos de RCGA diferentes, um local e o outro global. O RCGA local atinge soluções mais precisas quando se trabalha com problemas unimodais, enquanto que o RCGA global atinge soluções mais confiáveis quando se trabalha com problemas multimodais e complexos. Também deve ser destacado que o RCGA local utiliza baixos valores de N_f, enquanto que o RCGA global faz uso de altos valores desse grupo de cromossomos femininos. Ainda acerca desse algoritmo, deve-se observar que os melhores indivíduos da população final do RCGA global, passam a ser a população inicial do RCGA local. Desta forma, o RCGA global é executado primeiro, em busca de regiões promissoras, enquanto que o RCGA local é executado logo após, a partir dessas áreas promissoras apontadas pelo RCGA global, e intensifica a busca nessas regiões para encontrar resultados mais precisos.

Para a competição do CEC 2005, este algoritmo foi configurado com os seguintes parâmetros:

- $P_G = 50\%$;
- O RCGA global utiliza o operador de cruzamento PBX-α com $\alpha = 0.8$, $\mu = 2$ e $\lambda = 1$. Além disso, $N_F = 100$ e $N_M = 400$;
- O RCGA local aplica o operador de cruzamento PCX com $\sigma_\varsigma^2 = 0.1$, $\sigma_\eta^2 = 0.1$, $\mu = 3$ e $\lambda = 2$. Além disso, $N_F = 1$ e $N_M = 200$.

3.2 BLX-MA

Conforme os estudos de Molina, Herrera e Lozano (2013), o nome do algoritmo BLX-MA foi dado pela junção de *Blended Crossover* (BLX) com *Memetic Algorithm* (MA). Como o nome já sugere, ele é uma combinação entre um algoritmo genético e um algoritmo de busca local, e quando se faz essa combinação, obtém-se o que se chama de Algoritmo Memético. Como ele combina dois algoritmos em um, ele é, portanto, um algoritmo híbrido, assim como o anterior, o BLX-GL50. Esses dois algoritmos são bem parecidos entre si, pois os dois fazem uso de um algoritmo genético do tipo global RCGA para promover uma exploração global, porém o BLX-GL50 utiliza outro algoritmo genético, o RCGA local, para promover uma exploração local, enquanto que o BLX-MA utiliza um algoritmo de Busca Local para realizar esta tarefa de exploração por intensificação (*exploit*).

Quando se trabalha com problema de otimização de função em espaço de busca contínuo (domínio contínuo), um detalhe deve ser levado em conta: soluções com maior grau de precisão devem ser obtido com algoritmos de resolução específicos, os chamados *solvers*. O algoritmo de busca local LS (*Local Search*) é um *solver*, e o papel dele, ao ser incorporado a um RCGA, é refinar eficientemente as soluções propostas pelo RCGA.

Como foi dito anteriormente, ao se combinar um algoritmo LS com um Algoritmo Genético, obtém-se um Algoritmo Memético que, assim como os algoritmos genéticos, pode ser do tipo *real-coded*, que permite trabalhar com o espaço de busca de uma maneira mais natural do que o alfabeto binário, que é o método mais tradicional usado pelos algoritmos genéticos. Ao fazer essa

combinação de um RCGA com um LS, tem-se um RCMA (*real-coded memetic algorithm*), e o BLX-MA é exatamente isso: um RCMA.

Uma formulação comumente usada pelos algoritmos meméticos aplica a busca local aos membros da população após sofrerem recombinação ou mutação, com o objetivo de intensificar a busca por soluções nas melhores regiões obtidas através da execução do RCGA. O BLX-MA funciona dessa forma, ou seja, como um algoritmo memético qualquer, com a diferença que ele ajusta os parâmetros Probabilidade e Profundidade do algoritmo de busca local. Ele usa a aptidão dos indivíduos da população para decidir quando o LS deve ser aplicado (probabilidade) e quanto esforço deve ser gasto por ele (profundidade), focando sempre todo o esforço nas regiões mais promissoras.

O RCGA utilizado pelo BLX-MA, possui as seguintes características:

- Trabalha com o operador de cruzamento BLX-0.5, que é o mesmo BLX-α, quando tem-se $\alpha = 0.5$. Isso garante uma significante diversidade à população gerada por esse RCGA, e isso evita uma situação de falha conhecida dos RCGAs: a Convergência Prematura. Esse problema faz com que a busca pela solução não tenha um bom alcance, o que pode prender o algoritmo a um ótimo local em vez de permitir que ele encontre um ótimo global;
- Aplica a mutação não-uniforme nos seus cromossomos;
- Faz uso do NAM (*Negative Assortive Mating*) como método de seleção, assim como o BLX-GL50;
- Utiliza a estratégia RW (replace worst), que substitui o pior indivíduo por um novo se e somente se o novo indivíduo for melhor do que o atual, como ocorre também no BLX-GL50.

O BLX-MA também faz uso de um algoritmo de busca local, o LS, e esse algoritmo opera tipicamente em uma pequena porção do total de soluções visitadas, pois suas funções de avaliação adicionais possuem um custo computacional muito elevado. Então, foi introduzido ao BLX-MA um parâmetro chamado de Probabilidade LS (P_{LS}), que é responsável por determinar quando o LS deve ser invocado para refinar um novo cromossomo. A questão que surge naturalmente desta situação é como melhor selecionar as soluções que serão submetidas ao LS.

Utilizando o conceito de “farejar”, em que um animal sente o cheiro da comida quanto mais próximo ele estiver dela, o BLX-MA também prevê que existem boas soluções próximas à melhor solução. Desta forma, deve ser alocado mais recurso computacional para as regiões mais promissoras do que para as menos promissoras, pois o “cheiro” de solução ótima é maior nas regiões mais promissoras.

Sabe-se que existem dois mecanismos diferentes para se calcular adaptativamente a aplicação do LS:

- Baseada na aptidão: acredita-se que indivíduos com maior aptidão tendem a estar mais próximos da solução ótima, e por conta disso, ela modifica a P_{LS} baseando-se na relação entre a aptidão de um indivíduo e a do resto da população;
- Baseada na distribuição: acredita-se que para fugir de ótimos locais e encontrar-se o ótimo global, deve-se ir em busca das soluções mais distantes entre si, modificando-se a P_{LS} de maneira favorável às regiões mais longínquas (distribuídas) entre si.

Como o RCGA entrega sua solução global baseado na população, é natural que o BLX-MA utilize o primeiro método, baseado na distribuição dos indivíduos com maiores aptidões apontados pelo RCGA global.

Agora que já foi compreendido como funcionam o RCGA e o LS, a proposta dos autores do BLX-MA é conceitualmente simples: deve-se classificar a prole em 3 categorias, de acordo com a aptidão (fitness = f), e atribuir diferentes valores de P_{LS} e número de iterações do LS (N_{itera}) para cada categoria. Esses valores são exibidos no quadro 02 a seguir:

Quadro 02 – Aplicação do LS

Grupo	quando	P_{LS}	N_{itera}
Mais promissores	$f(new) > f(best)$	100%	100
Medianos	$f(best) \geq f(new) \geq f(worst)$	6,25%	50
Menos promissores	$f(new) < f(best)$	0%	-

Fonte: adaptado de Molina, Herrera e Lozano (2013, p. 889)

Na primeira categoria têm-se indivíduos com aptidão melhor do que a melhor aptidão na população atual, e eles são identificados como as soluções mais

promissoras e por isso o LS deve ser aplicado 100 vezes em cada indivíduo desse grupo: P_{LS} = 1.0 (100%) e N_{itera} = 100. Já na segunda categoria, têm-se os indivíduos com aptidão entre a melhor e a pior aptidões da população são considerados elementos promissores e que merecem um ajuste local de 50 iterações do LS, mas com menor probabilidade do que os da primeira categoria: P_{LS} = 0.00625 (6.25%, valor padrão) e N_{itera} = 50. Na terceira categoria, onde se tem os indivíduos com aptidão pior do que o pior da população, esses indivíduos são considerados muito ruins para se gastar recurso computacional e, portanto, não devem ser submetidos ao LS: P_{LS} = 0 (0%).

3.3 CoEVO

Pošík (2013) relata em seus estudos que o algoritmo CoEVO é um algoritmo evolutivo para otimizar funções com parâmetros reais. Essa otimização com parâmetros reais é uma importante questão em muitas áreas da atividade humana e por conta disso, foram criados diversos algoritmos para solucionar essa tarefa, indo desde o simples *Hill Climbing* (Escalada da Encosta) até os complexos algoritmos baseados em população. Essa tarefa é uma típica área de aplicação para as Estratégias Evolutivas – *Evolutionary Strategies* (ES), em inglês.

As ES evoluem não apenas os vetores de solução, mas também os parâmetros de distribuição do seu vetor de mutação. Cada indivíduo tem sua própria distribuição de mutação. Recentemente, uma estratégia evolutiva com adaptação da matriz de covariância (CMA-ES) se tornou muito popular, pois ela é uma ferramenta muito eficiente para otimização com parâmetros reais. Nessa estratégia, uma nova população é criada por diversas mutações da população atual, e após uma avaliação, a população da prole é usada para atualizar a população atual e a matriz de covariância. Desta forma, passos de mutação são reproduzidos com sucesso.

O algoritmo CoEVO é uma estratégia evolutiva porém, ele não faz uso de nenhuma matriz de covariância. Em vez disso, ele co-evolui a população de etapas de mutação bem sucedidas de gerações anteriores. Essa população substitui o modelo probabilístico (as variâncias ou matrizes de covariância) usado nas ES. A nova população é então criada pela mutação de todos os membros da população antiga, o que favorece que o algoritmo realize a pesquisa em diversas regiões do

espaço de busca, e não apenas na região coberta pela nuvem gaussiana, como no caso da CMA-ES.

O comportamento desse algoritmo também lembra os algoritmos de otimização por enxame de partículas, os PSO (*Particle Swarm Optimization*). Estes também adaptam suas etapas de mutação através dos seus vetores de velocidade, mas fazem uso do histórico de cada indivíduo além da informação global, enquanto que o CoEVO não tira proveito da relação entre as partículas e os vetores de velocidade do indivíduo. No CoEVO, diz-se que as etapas de mutação não são "propriedade" dos indivíduos.

Essa característica também determina o conjunto para o qual este algoritmo é bem adequado. O panorama de aptidão deve favorecer que a etapa de mutação mais recente seja bem sucedida também para mutações futuras, ou seja, para outros membros da população situados em diferentes regiões do espaço de busca. Esta condição é satisfeita para:

- Funções unimodais;
- Funções multimodais com panoramas de aptidão que se encontram nas vizinhanças do ponto ótimo, semelhantes entre si.

Isto não quer dizer, porém, que caso estas condições não sejam satisfeitas, o algoritmo seja incapaz de resolver o problema em questão. Apenas ocorre que a vantagem da cooperação através do compartilhamento de etapas de mutação irá desaparecer, e desta forma o algoritmo irá se comportar como uma estratégia com mutações randômicas, ou seja, as etapas de mutações dos indivíduos não irão mais cooperar umas com as outras, e sim, disputar entre si.

O algoritmo CoEVO opera com duas populações: a primeira delas é a população de soluções potenciais da geração atual, e a segunda é a população de etapas de mutação (por exemplo, população de vetores de mutação) que foram bem sucedidas em gerações anteriores. O algoritmo do CoEVO pode ser descrito brevemente no quadro 03 que segue:

Quadro 03 – Algoritmo CoEVO

```
Initialize both populations;
Evaluate the population of solutions;
Repeat until the stopCriterion is fullfilled{
    Create a set of candidate solutions and related candidate mutation steps;
    Evaluate candidate solutions and mutations steps;
    Select better candidate solutions and mutation steps to compete for survival;
    Replace worse population members by selected candidate solutions and mutation steps;
}
```

Fonte: adaptado de Pošík (2013, p. 872-873)

Ao executar este algoritmo, a população de soluções evolui através de sucessivas mutações, e cada etapa de mutação bem-sucedida é armazenada na população de etapas de mutação. Cada mutação ocorre selecionando-se os elementos da população de soluções atual, e aplicando a eles uma das etapas de mutação selecionada de forma randômica na população de etapas de mutação. A população de etapas de mutação sempre é incrementada a cada iteração do algoritmo, pois a primeira mutação ocorre na realidade, de uma forma bem semelhante a um cruzamento, já que dois membros da população de soluções são selecionados, de forma aleatória, para virarem genitores de um novo indivíduo que é escolhido randomicamente em uma reta imaginária entre esses dois genitores. Esse valor é inserido como um novo membro da população de etapas de mutação.

Após o término do processo de mutação, obtém-se uma nova população de soluções candidatas, que é avaliada juntamente com a população atual, e a mais vantajosa das duas é mantida enquanto a outra é descartada. Isso ocorre até que não haja mais substituição de nenhum membro da população atual por outro da população candidata. O critério utilizado para avaliação dos indivíduos é a aptidão.

Em suma, o que acontece na execução desse algoritmo é que as duas populações cooperam entre si para buscar a solução ótima, e essas duas populações evoluem paralelamente (coevoluem) durante a execução do algoritmo.

3.4 DE

Segundo Rönkkönen, Kukkonen e Price (2013), o algoritmo DE (*Differential Evolution*) – Evolução Diferencial, em português – foi desenvolvido por

Rainer Storn e Kenneth Price. Ele é um otimizador estocástico de funções, baseado em uma população de possíveis soluções, simples, mas ao mesmo tempo muito poderoso. Ele trabalha com uma população de possíveis soluções e utiliza um vetor para perturbar essa população e tentar encontrar uma solução melhor do que a atual.

Este algoritmo trabalha com os seguintes parâmetros de controle:

- D: número de dimensões (tamanho do cromossomo);
- N_P: tamanho da população;
- C_R: constante de *crossover* (cruzamento).

O algoritmo do DE é mostrado no quadro 04 a seguir:

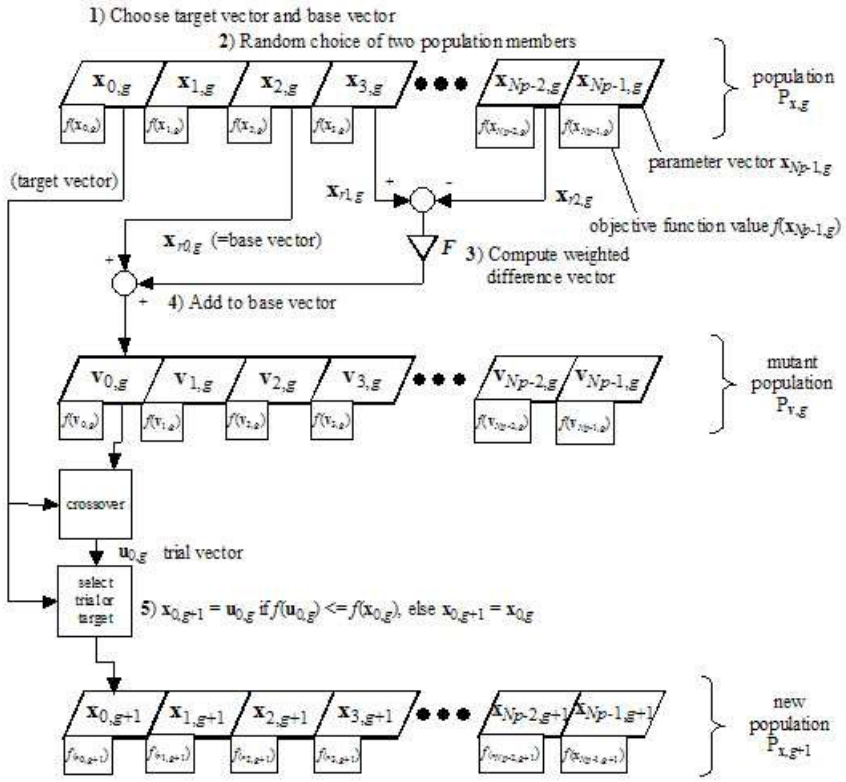
Quadro 04 – Algoritmo DE

```
Inicializar randômica e uniformemente a população de possíveis soluções;
Avaliar a população;
Atribuir o valor 0 à geração;
Repetir até que se satisfaça o critério de parada{
    Selecionar randomicamente um vetor alvo;
    Selecionar randomicamente um vetor base; (genitor principal)
    Selecionar randomicamente dois vetores genitores; (genitores secundários)
    Realizar diferenciação entre os vetores genitores;
    Efetuar mutação no vetor base utilizando o resultado da diferenciação;
    Avaliar o vetor mutante resultante da mutação;
    Realizar o cruzamento entre o vetor mutante e o vetor alvo;
    Avaliar o vetor experimental resultante do cruzamento;
    Selecionar o melhor entre o vetor alvo e o vetor experimental;
    Avaliar a população;
    Incrementar o valor da geração em 1 unidade;
}
```

Fonte: adaptado de Rönkkönen, Kukkonen e Price (2013); Price e Storn (2013, não paginado); Boccato, Attux e Zuben (2009);

Complementando o quadro 04, a figura 03 que segue exibe o comportamento do algoritmo DE:

Figura 03 – Comportamento do DE



Fonte: Price e Storn (2013, não paginado)

Em outras palavras, este algoritmo inicia com a criação da população N_P de forma aleatória, porém dentro de limites superior e inferior pré-estabelecidos pela pessoa. Cada indivíduo (vetor) desta população possui D características (informações). É atribuído o valor 0 (zero) à geração, para indicar que esta é a geração inicial da execução do algoritmo.

Em seguida escolhem-se 4 vetores de forma randômica, sendo um o vetor alvo, que é armazenado para posterior comparação, e os outros três são os vetores genitores, sendo que um deles é o genitor principal e os outros dois são os genitores secundários. Efetua-se o processo de diferenciação entre os genitores secundários, em cada uma das suas D características, e logo em seguida é feita uma mutação no genitor principal, utilizando-se também cada uma das suas características, dando origem a um vetor mutante também com D características.

Logo após, realiza-se um cruzamento entre este vetor mutante e o vetor alvo selecionado no início da execução do algoritmo, e obtém-se um vetor resultante chamado de vetor experimental. Este vetor é comparado ao vetor alvo, e o que tiver

a melhor aptidão (fitness) é mantido e o outro é removido da população, e logo após uma mudança na população, a geração deve ser incrementada em 1 unidade.

Este algoritmo é designado a resolver qualquer tipo de problema, pois cada um dos seus parâmetros pode ser adaptado de acordo com o interesse de cada um que faça uso do mesmo. Por conta disso e para se caracterizar as principais variantes de algoritmos DE, foi criada uma nomenclatura para ele: DE/x/y/z; onde x representa o vetor a ser substituído, y representa o número de operações de diferenciação utilizadas no processo de mutação, e z representa o critério de cruzamento utilizado pelo algoritmo. O algoritmo que competiu no CEC 2005 foi o DE padrão, também conhecido por *classic* DE, ou seja, DE/rand/1/bin. Como quase sempre o critério de cruzamento adotado é o binário, ele pode ser omitido da nomenclatura do DE, que automaticamente o DE mencionado é interpretado como um DE com cruzamento binário, como por exemplo, em DE/rand/1.

3.5 DMS-L-PSO

Liang e Suganthan (2013) explanam em seus estudos sobre o algoritmo DMS-L-PSO. O mesmo tem seu nome dado pela abreviatura de *Dynamic Multi-Swarm Particle Optimizer with Local Search*, que, como o nome já diz, é um algoritmo híbrido que reúne um algoritmo Otimizador de Enxame de Partículas com Multi-enxames Dinâmicos (DMS-PSO), e um algoritmo de Busca Local (LS).

O PSO, desenvolvido por Kennedy e Ebenhart, emula o comportamento de pássaros migratórios para solucionar problemas de otimização. No PSO, cada solução potencial é considerada como uma partícula. Todas as partículas possuem valores de aptidão e velocidades. A partícula “voa” (move-se) através do espaço D-dimensional do problema orientando-se pelo histórico de informações de todas as partículas. Utilizando as úteis informações obtidas no processo de busca, as partículas tendem a voar em direção às melhores regiões de pesquisa ao longo do processo de busca.

Diferentemente de outros PSOs multi-enxames e versões locais de PSOs, no DMS-L-PSO os enxames são dinâmicos e pequenos. Toda a população é dividida em vários enxames pequenos, e esses enxames são frequentemente reagrupados utilizando vários itinerários de reagrupamento e as informações são

compartilhadas entre os exames. Além disso, ele incrementa a habilidade do seu algoritmo Busca Local combinando-o com o método Quasi-Newton.

Existem duas versões de PSO puros, sendo uma global e a outra local. A principal diferença entre eles é que na versão local, cada velocidade de cada partícula é ajustada de acordo com sua melhor performance pessoal e a melhor performance mais distante atingida dentro de sua vizinhança, enquanto que na versão global, o ajuste da velocidade de cada partícula é feito de acordo com a melhor performance pessoal da partícula e a melhor performance mais distante atingida por toda a população. O DMS-PSO baseia-se na versão local, porém ele diferencia-se por usar multi-enxames, cuja topologia da vizinhança é dinâmica e randomicamente especificada. Isso confere ao DMS-L-PSO uma melhor performance em problemas multimodais, porém a sua performance local não é muito boa. Para corrigir esse problema que são adicionados ao DMS-PSO, o LS e o método Quasi-Newton.

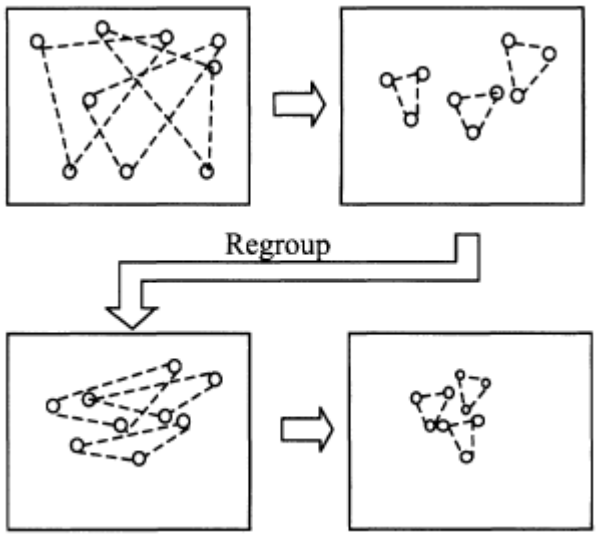
O DMS-L-PSO utiliza população pequena, pois sabe-se que com uma população de 3 a 5 elementos, consegue-se atingir resultados satisfatórios para problemas simples. Além disso, também se têm relatos de que em PSO locais, os que trabalham com vizinhança pequena possuem melhor desempenho com funções complexas. Porém, isso pode causar problema de convergência prematura e desta forma, o algoritmo pode ficar preso um ótimo local, em detrimento de continuar a busca pelo ótimo global. Sendo assim, o DMS-L-PSO aumenta um pouco o tamanho da população, porém divide ela em enxames menores, com poucos membros, e cada enxame usa seus próprios membros para buscar as melhores regiões no espaço de busca.

Uma vez que os pequenos enxames do DMS-L-PSO estão pesquisando pelas melhores regiões, é fácil eles convergirem para um ótimo local graças à propriedade de convergência do seu PSO. Isso torna ele bem parecido com um PSO evolutivo, nos quais os enxames também fazem buscas paralelas entre si, porém a diferença é que no DMS-L-PSO as informações de cada partícula são compartilhadas entre os enxames, já que de tempo em tempo os enxames são reagrupados de forma randômica. Esta é a principal idéia por detrás deste algoritmo.

Um exemplo deste funcionamento pode ser descrito por uma população de 9 partículas buscando otimizar um problema. Essas 9 partículas são divididas em 3 enxames com 3 partículas cada. Cada enxame passa a buscar por ótimos locais

utilizando seus membros, e isso faz com que eles comecem a convergir para regiões próximas a esses ótimos locais. Após certo tempo, os enxames são reagrupados randomicamente, e cada membro (partícula) leva suas informações para o novo enxame, fazendo com que os enxames compartilhem suas informações entre si, como pode ser visto na figura 04 que segue:

Figura 04 – DMS-L-PSO Search



Fonte: Liang e Suganthan (2013, p. 523)

O algoritmo DMS-L-PSO trabalha com os seguintes parâmetros:

- m: tamanho da população de cada enxame;
- n: número de enxames;
- R: período de reagrupamento;
- L: período de refinamento local;
- L-FEs: Máximo de Avaliações da Função usando Busca Local (LS);
- Max_FEs: Valor máximo de avaliações de *fitness* (condição de parada).

O algoritmo é descrito no quadro 05 a seguir:

Quadro 05 – Pseudocódigo do DMS-L-PSO

Initialize $m*n$ particles (position and velocity)
Divide the population into n swarms randomly, with m particles in each swarm.
FEs=0; gen=0

While $FEs < 0.95*Max\_FEs$
gen=gen+1;
For i=1:$m*n$
  Find $lbest_i$
  For d=1:D
    If rand<0.5

$$V_i^d = w*V_i^d + c_1*rand1_i^d*(pbest_i^d - X_i^d)$$
$$+c_2*rand2_i^d*(lbest_i^d - x_i^d)$$
$$V_i^d = \min(\max(V_i^d, -V_{\max}^d), V_{\max}^d)$$
$$X_i^d = X_i^d + V_i^d$$

    Else
$$X_i^d = pbest_i^d$$
    End
  End

  If $X_i \in [X_{\min}, X_{\max}]^D$
    Calculate the fitness value
    FEs= FEs+1
    Update $pbest$
  End
End

If mod (gen, L)==0,
  Sort the $lbest$ according to their fitness value and refine the first $\lceil 0.25n \rceil$ best $lbest$ using Quasi-Newton method.
  $FEs = FEs + \lceil 0.25n \rceil * L\_FEs$
  Update the corresponding $pbest$
End

If mod (gen, R)==0,
  Regroup the swarms randomly,
End
End

Refine the best solution achieved so far using Quasi-Newton method with $0.05*Max\_FEs$ fitness evaluations.

Fonte: Liang e Suganthan (2013, p. 524)

Os ajustes dados a este algoritmo para a competição no CEC 2005 foram $n = 20$ e $m = 3$ (totalizando a população em 60 partículas), w = 0,729, $c_1 = c_2 = 1{,}49445$, $R = 10$, $L = 100$, $L\_FEs = 200$. Max_FEs = 100.000 para 10-D e 300.000 para 30-D. V_{max} restringe a velocidade das partículas, já que ele é configurado para 20% do valor do alcance da busca.

3.6 EDA

O EDA, sigla para *Estimation of Distribution Algorithm*, é um singular algoritmo evolutivo, muito simples, baseado na modelagem probabilística em detrimento dos operadores genéticos como *crossovers* e mutações, e foi desenvolvido por Yuan e Gallagher (2013). O mecanismo fundamental deste algoritmo é conduzir a pesquisa por meio de amostragem de novos indivíduos a partir de uma distribuição de probabilidade, que é estimada com base em alguns

indivíduos promissores selecionados na população atual. A principal vantagem dos EDAs é que eles podem explicitamente aprender as dependências entre variáveis do problema e usar essa informação estrutural para gerar eficientemente novos indivíduos.

O EDA utilizado na competição do CEC 2005 é contínuo, simples e utiliza a distribuição Gaussiana multivariada para modelar indivíduos selecionados e gerar novos indivíduos. Este EDA possui algumas pequenas modificações e é referenciado como EDA_{mvg}, onde MVG significa *MultiVariate Gaussian Model*. Seu pseudocódigo é mostrado no quadro 06 que segue:

Quadro 06 – Pseudocódigo do EDA_{mvg}

```
Initialize and evaluate the population P
  While stopping criteria not met
    Select some individuals P^sel from P.
    Estimate the mean μ and covariance Σ of P^sel.
    Sample a population P' from G (μ, Σ).
    Evaluate individuals in P'.
    Combine P and P' to create a new P.
  End While
```

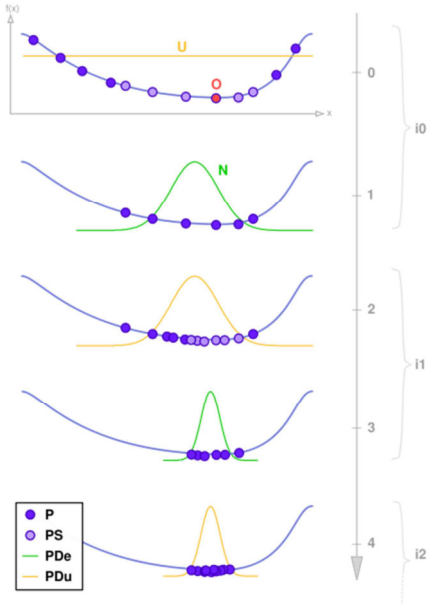
Fonte: Yuan e Gallagher (2013, p. 1792)

O procedimento geral do EDA_{mvg} é estimar, em cada geração, as estatísticas μ e Σ de indivíduos promissores selecionados da população atual e, gerar novos indivíduos P' por amostragem da Gaussiana correspondente de modo que os novos indivíduos sigam a mesma distribuição assim como os promissores.

Quando este algoritmo tem sua execução iniciada, a população inicial é gerada randomicamente dentro do espaço de busca. Então, o EDA_{mvg} tipicamente inicia com um desvio padrão relativamente elevado, o que lhe confere a capacidade de promover a busca em uma grande área e o torna menos provável a estagnar em um ótimo local, quando comparado a métodos de busca local. Por outro lado, uma vez que ele identifica uma região promissora a conter o ótimo global, ele pode rapidamente reduzir seu escopo de busca, representado pelos valores próprios da matriz de covariância, para alcançar uma rápida velocidade de convergência.

A figura 05 a seguir demonstra seu comportamento quando está realizando a tarefa de otimizar um determinado problema:

Figura 05 – Comportamento do EDA



Fonte: Wikipedia (2013b, não paginado)

Uma deficiência inerente do EDA_{mvg} é que ele não lida eficientemente com problemas multimodais, o que é ocasionado principalmente devido à única distribuição Gaussiana em uso. Porém, isto não quer dizer necessariamente que o EDA_{mvg} nunca irá trabalhar bem com estes problemas. Pesquisas anteriores provaram que enquanto o problema exibe a estrutura "*Big Valley*", o EDA_{mvg} ainda pode ter uma boa chance de sucesso.

Os autores Yuan e Gallagher (2013) também noticiaram em seus estudos que foram identificadas algumas questões adicionais acerca do EDA_{mvg}, que podem implicar em uma significante perda de performance. Primeiramente, foi noticiado que este algoritmo pode ficar estagnado prematuramente em um ótimo local em problemas unimodais como a função esfera (*Sphere function*), especialmente em casos com muitas dimensões. A situação ideal para o bom desempenho deste algoritmo é quando a Gaussiana está exatamente sobre o ótimo global, pois isto faz com que o algoritmo encontre a resposta rapidamente e com um consistente escopo de busca. Porém, quando o vetor média (μ) não está próximo ao ótimo global, seria necessário que o algoritmo realizasse uma "escalada" em busca do ótimo global,

como faz o algoritmo *Hill Climbing*, mas isso não ocorre, pois estudos acerca da dinâmica deste algoritmo apontaram que os valores mais elevados da matriz, que são usados para gerar os novos indivíduos da população, rapidamente tendem a zerar, enquanto o vetor continua distante do ótimo global e não atinge o mesmo. A chave para corrigir esse problema é manter a diversidade da população de forma explícita, porém isto incorre em outra questão: quando isto deve ser feito, pois caso o topo da Gaussiana esteja próxima ao ótimo global, aumentar a diversidade da população irá acarretar lentidão na convergência do algoritmo, e consequentemente irá necessitar mais execuções e pode ultrapassar o limite de tempo ou o de execuções. Uma heurística simples adotada para solucionar esse caso é verificar a distância entre o vetor média e o melhor indivíduo na população. Caso o melhor indivíduo esteja em uma distância inferior a um determinado limite, do vetor média, isto para cada dimensão, então nesse caso não é necessário aplicar a manutenção da diversidade, pois é muito provável que a Gaussiana esteja próxima ao ótimo global. Caso essa distância seja maior do que o limite predefinido, então o algoritmo ainda deve continuar a busca pelo ótimo global e não deve convergir tão rapidamente. Além disso, também deve ser levado em conta como manter a diversidade. Uma forma simples é ampliar a matriz de covariância em um fator maior do que 1. Outro modo mais preciso para manter a variedade, é tratar cada dimensão de modo separado, pois o melhor indivíduo pode estar próximo ao ótimo global naquela dimensão, mas distante dele em outra dimensão, porém isso consome mais recurso computacional e por este motivo, o algoritmo que competiu usou a técnica de elevar os valores próprios da matriz.

Segundo, foi visto que com a ajuda da técnica de manutenção de diversidade, o desempenho do EDA_{mvg} pode ser melhorada para alguns problemas, porém a velocidade de convergência não é rápida o suficiente para permitir que se encontre uma solução muito boa dentro de um número limitado de execuções (avaliações de *fitness*).

Terceiro, sabe-se que para alguns problemas existe um grande número de ótimos locais ao redor do ótimo global, com qualidades comparáveis à do mesmo. Neste caso, os indivíduos selecionados estão distribuídos em uma região muito ampla, fazendo com que a matriz de covariância possua muitos valores principais, o que pode prevenir que as melhores soluções sejam encontradas eficientemente. Para corrigir isso, a população deveria ser dividida em blocos, e

cada bloco deveria ter sua própria Gaussiana, porém isso acarretaria em um consumo muito elevado de recursos computacionais, para poder lidar com um grande e incerto número de blocos, e isso poderia acarretar em exceder os limites definidos pela competição do CEC 2005. Então, para solucionar esta questão, a técnica utilizada foi elevar a pressão de seleção, ou seja, forçar o algoritmo a selecionar os indivíduos realmente muito bons, os melhores entre os melhores, para manter uma boa velocidade de convergência e ao mesmo tempo convergir para o ótimo global.

Os parâmetros de controle deste algoritmo são:

- P: tamanho da população;
- Q: valor máximo de amplificação (deve ser maior ou igual a 1);
- α: taxa de aprendizagem (deve ser maior ou igual a 0 e menor ou igual a 1);
- τ: proporção de seleção (deve ser maior que 0 e menor ou igual a 1).

Desses parâmetros, apenas P faz parte do algoritmo EDA original. Os outros três parâmetros foram adicionados ao EDA_{mvg} utilizado no comparativo a fim de solucionar as 3 questões citadas anteriormente. Os parâmetros foram configurados para as execuções com 10-D com os seguintes valores: P=200; Q=2,0; α=0,2 e τ=0,3. Para 30-D, os valores adotados foram: P=1.000; Q=1,5; α=0,2 e τ=0,3. Com o intuito de melhorar ainda mais o desempenho deste algoritmos nos problemas de 6 a 10 do CEC 2005, foram modificados os seguintes parâmetros:

- α=0, para o problema número 6 (isto é, sem aprendizado incremental);
- Q=1,0 e τ=0,2, para os problemas de números 9 e 10 (isto é, pressão de seleção muito forte e sem manutenção de diversidade da população).

3.7 IPOP-CMA-ES (ou G-CMA-ES)

O IPOP-CMA-ES (também conhecido como G-CMA-ES), descrito nos trabalhos de Auger e Hansen (2013) é um algoritmo variante do algoritmo CMA-ES original, com a adição da regra de reiniciar o algoritmo em certas condições estabelecidas, e a principal mudança é o incremento da população a cada reinício.

CMA-ES é uma sigla que representa *Covariance Matrix Adaptation Evolution Strategy*, e o CMA-ES é uma estratégia evolutiva que adapta toda a matriz de covariância de uma distribuição de busca normal (mutação). Uma característica peculiar do CMA-ES em relação a outros algoritmos evolutivos é que ele não varia perante transformações lineares do espaço de busca: o CMA-ES exibe a mesma performance em uma dada função objetivo $f: x \in \mathbb{R}^n \to f(x) \in \mathbb{R}$, onde $n \in \mathbb{N}$, e nessa mesma função com uma transformação linear aplicada a ela, como ocorre em $f_R: x \in \mathbb{R}^n \to f(Rx) \in \mathbb{R}$, onde R representa uma transformação linear, $\mathbb{R}$ o conjunto dos números Reais e $\mathbb{N}$ o conjunto dos números Naturais. Isto ocorre somente se for feita uma transformação correspondente dos parâmetros da estratégia (distribuição).

O CMA-ES mais comumente utilizado é denominado por $(\mu/\mu_W, \lambda)$-CMA-ES, onde as variáveis de controle são *μ*, que é a média da população de melhores soluções; *w*, que é o peso atribuído à média da população para efeito de cálculos; e *λ*, que representa a quantidade de novas soluções candidatas. O funcionamento deste algoritmo é composto de 3 partes principais, que são:

1) gerar uma amostra de novas soluções;

2) reordenar as soluções da amostra baseado em seus *fitness*;

3) atualizar as variáveis de estado interno baseado nas amostras que foram reordenadas do passo anterior.

No quadro 07 a seguir, é exibido o pseudocódigo do CMA-ES:

Quadro 07 – Pseudocódigo do CMA-ES

```
set λ  // number of samples per iteration, at least two, generally > 4
initialize m, σ, C = I, p_σ = 0, p_c = 0  // initialize state variables
while not terminate  // iterate
   for i in {1...λ}  // sample λ new solutions and evaluate them
      x_i = sample_multivariate_normal(mean=m, covariance_matrix=σ^2 C)
      f_i = fitness(x_i)
   x_{1...λ} ← x_{s(1)...s(λ)} with s(i) = argsort(f_{1...λ}, i)  // sort solutions
   m' = m  // we need later m − m' and x_i − m'
   m ← update_m(x_1, ..., x_λ)  // move mean to better solutions
   p_σ ← update_ps(p_σ, σ^{-1} C^{-1/2} (m − m'))  // update isotropic evolution path
   p_c ← update_pc(p_c, σ^{-1} (m − m'), ||p_σ||)  // update anisotropic evolution path
   C ← update_c(C, p_c, (x_1 − m')/σ, ..., (x_λ − m')/σ)  // update covariance matrix
   σ ← update_sigma(σ, ||p_σ||)  // update step-size using isotropic path length
return m or x_1
```

Fonte: Wikipedia (2013c, não paginado)

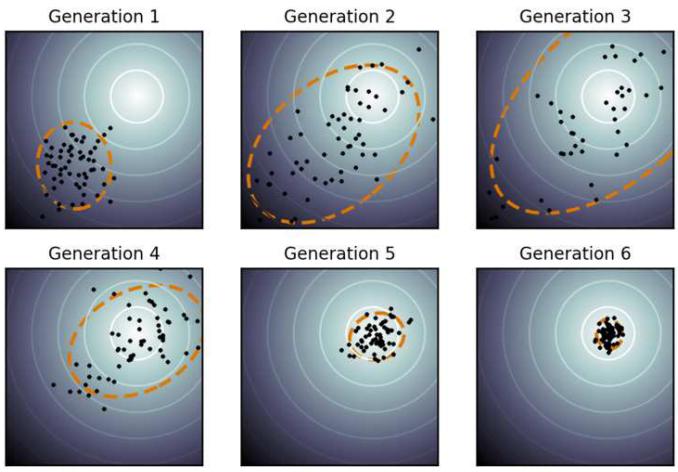
Além das variáveis citadas anteriormente, também se tem *n*, que é a dimensão do espaço de busca e *k*, que é o passo da iteração. Além destas, tem-se também as variáveis de estado que são:

- $m_k \in \mathbb{R}^n$, que é a média da distribuição e a atual solução favorita para a otimização do problema;
- $\sigma_k > 0$, que é o tamanho do passo;
- C_k, que é uma simétrica, positiva e definida matriz de covariância *n* x *n*, com $C_0 = I$;
- $P_\sigma \in \mathbb{R}^n$, $P_c \in \mathbb{R}^n$, que são dois caminhos evolutivos, inicialmente configurados como o vetor zero.

O CMA-ES otimiza eficientemente funções unimodais e é particularmente superior em problemas mal condicionados e não-separáveis. Hansen e Kern, desenvolvedores do CMA-ES, mostraram que incrementar o tamanho da população melhora o desempenho do CMA-ES em funções multimodais. Consequentemente, eles sugeriram uma estratégia de restart (reinício) de CMA-ES com sucessivos incrementos do tamanho da população. Este algoritmo foi referenciado como IPOP-CMA-ES.

Ainda acerca do CMA-ES, tem-se a figura 06 a seguir, a qual demonstra o comportamento do CMA-ES buscando o ótimo global na região de pesquisa de uma função. Deve ser notado que a população primeiro “se espalha” pela região de pesquisa, e depois vai se concentrando novamente em torno do ótimo global.

Figura 06 – Comportamento do CMA-ES



Fonte: Wikipedia (2013c, não paginado)

Uma das variações do CMA-ES é o restart-CMA-ES, também conhecido como R-CMA-ES, que é um CMA-ES com a adição de critérios que forçam o reinício do algoritmo, para melhorar a performance. O IPOP-CMA-ES é baseado no R-CMA-ES, ou seja, obedece aos mesmos critérios de reinício deste, porém a cada reinício, ele incrementa o tamanho da população em um fator de 2. Os critérios de reinício do algoritmo R-CMA-ES, e consequentemente também do IPOP-CMA-ES, são:

- Parar se o alcance dos melhores valores da função objetivo das últimas 10 + $\lceil 30n/ \rceil$ gerações é zero, ou o alcance desses valores da função e todos os valores da função das gerações recentes for abaixo de Tolfun $= 10^{-12}$;
- Parar se o desvio padrão da distribuição normal for menor que TolX em todas as coordenadas e P_C for menor do que Tolx em todos os componentes, onde Tolx $= 10^{-12}$;
- Parar se ao adicionar um vetor de desvio padrão de 0,1 à direção do eixo principal de C^g, não afetar a solução candidata da geração atual, onde g é a geração;
- Parar se ao adicionar um desvio padrão de 0,1 em cada coordenada, não afetar a solução candidata da geração atual;
- Parar se o número de condição da matriz de covariância exceder o valor 10^{-4}.

Essa característica de aumentar a população a cada reinício do algoritmo concede ao mesmo uma significante melhora de desempenho de busca global, ou seja, a cada reinício do algoritmo, suas chances de encontrar o ótimo global sempre aumentam. Esse talvez tenha sido o principal motivo desse algoritmo ter se saído tão bem ao otimizar todas as funções do CEC 2005.

Para a competição do CEC 2005, o algoritmo IPOP-CMA-ES foi configurado com os valores padrão do CMA-ES original, exceto pelo número de dimensões, que no caso é definido de acordo com o problema, e no caso do CEC 2005 foram D= 10, 30 e 50. Este algoritmo demonstrou um ótimo desempenho em todas as 25 funções-teste do CEC 2005.

3.8 K-PCX

O algoritmo K-PCX é um algoritmo desenvolvido por Ankur Sinha, Santosh Tiwari e Kalayanmoy Deb, e é um baseado no algoritmo-gerador baseado em população. Infelizmente Sinha, Tiwari e Deb (2013) não espeficicaram em seus estudos o significado da letra K no nome do algoritmo, mas o PCX sabe-se que é de *Parent-Concentric Crossover*, que é o operador de cruzamento adotado por este algoritmo. Por conta disto, o K-PCX torna-se um singular competidor, pois o algoritmo-gerador baseado em população nada mais é do que um modelo genérico de como deve ser composto qualquer algoritmo de otimização baseado em população. Segundo o estudo dos autores do K-PCX, o algoritmo-gerador determina que 4 planos (*plans*) precisam ser especificados para gerar um procedimento de otimização do tipo steady-state, que são:

1) *Selection Plan* (SP): Plano de seleção, que é a escolha da estratégia utilizada para selecionar na população, um número fixo de genitores para realizar recombinação;
2) *Generation Plan* (GP): Plano de Geração, que consiste na metodologia usada para criar uma descendência de soluções a partir dos pais selecionados em SP;
3) *Replacement Plan* (RP): Plano de substituição, definido como a estratégia utilizada para selecionar um número fixo de membros (da população atual) que irão competir com as soluções da descendência gerada em GP, pela inclusão na população;
4) *Update Plan* (UP): Plano de atualização, que é a estratégia utilizada para decidir os vencedores de um conjunto de membros das soluções da descendência e que foram obtidos de RP, que serão eventualmente inseridos na população atual.

Sinha, Tiwari e Deb (2009) apontam ainda que esses planos foram desenvolvidos baseados nos aspectos essenciais necessários para resolver problemas de otimização unimodais e multimodais, como por exemplo a importância da preservação da diversidade, e a criação de uma descendência de soluções baseada na diversidade nas soluções dos genitores. Esse método (algoritmo-gerador) foi usado em outros esquemas de otimização de parâmetros-reais bem sucedidos, como ES e DE.

Com base nessas informações, Sinha, Tiwari e Deb simplesmente especificaram todos esses parâmetros do algoritmo-gerador baseando-se em estudos e experimentos anteriores, e fizeram uma afirmação muito interessante e que pode ser a peça chave que explica os resultados da competição do CEC 2005, que é abordada no capítulo 4 deste trabalho: os melhores resultados podem ser obtidos pelo ajuste fino dos parâmetros de controle para um problema particular. Eles afirmam ainda que como os problemas de teste do CEC 2005 englobam vários tipos de funções, como unimodais e multimodais, determinísticas ou com ruído, com baixas e altas dimensionalidades, etc., pode ser difícil que um único algoritmo de otimização consiga resolver todas elas em suas otimalidades. Isto ocorre porque para resolver um problema unimodal, uma determinada técnica pode ser boa, porém esta mesma técnica poder ser ruim para problemas multimodais, e será necessário adotar outra estratégia, e vice-versa. Outra coisa que também é acreditada por eles é que para resolver os variados “caprichos” dos problemas com um nível razoável de satisfação, o algoritmo deve ser simples e não deve ser projetado para resolver um problema em particular. De posse desse conhecimento, Sinha, Tiwari e Deb desenvolveram o K-PCX.

No K-PCX, o algoritmo é iniciado gerando-se randomicamente uma população de tamanho *N*, definido pelo usuário. A partir daí, usa-se um SP para escolher μ genitores dentro da população inicial. No esquema de seleção do K-PCX que competiu no CEC 2005, primeiro a população é ordenada de forma crescente, baseado no valor da função. Após isso, a população é dividida em *k* segmentos semelhantes, baseados no valor de *fitness*, onde *k* é um parâmetro definido pelo usuário, e que varia de 1 até *N*. A melhor solução de cada segmento é selecionada e armazenada em *B*, e então é selecionada randomicamente uma solução do conjunto de melhores soluções *B* para ser o primeiro genitor. Esse genitor é chamado de genitor *index* (índice) e seu índice é denotado por *p*. Consequentemente, os outros (μ - 1) genitores são escolhidos randomicamente na população.

No GP são criadas λ soluções descendentes baseadas nas soluções dos μ genitores, e então usa-se o operados de cruzamento PCX com modificações para o propósito de recombinação e produção de λ soluções descendentes. O operador PCX cria uma nova solução utilizando um número aleatório uniformemente distribuído, chamado de *u* (*u* está contido no intervalo [0,1]), de acordo com:

$$\vec{C} = \begin{cases} \vec{X}_p + \omega_\varepsilon \vec{D} + \sum_{i=1, i\neq p}^{\mu} \omega_\eta d\vec{e}_i, & \text{if}(u > 0{,}5\left(1 - \frac{1}{k}\right)) \\ \vec{M} + \omega_\varepsilon \vec{D} + \sum_{i=1, i\neq p}^{\mu} \omega_\eta d\vec{e}_i \end{cases}$$

Os termos usados na equação dada são definidos da seguinte forma:

- $\vec{X}_p \in B$ é o genitor índice;
- $\vec{G}$ é a média dos μ genitores;
- $\vec{D} = \vec{X}_p - \vec{G}$;
- $\vec{M}$ é a média de toda a população;
- d é a média das distâncias perpendiculares dos genitores $\vec{X}_i$ $(i \neq p)$ para a linha que une $\vec{X}_p$ e $\vec{G}$;
- $\vec{e}_i$ são os $(\mu - 1)$ vetores ortonormais que abrangem o subespaço perpendicular a $\vec{D}$.

Os dois parâmetros ω_ε e ω_η, usados para descrever a amplitude das variações na direção $\vec{D}$ e ortogonal a ele, respectivamente, são definidas da seguinte forma:

$$\omega_\varepsilon = k^{-\alpha} - \frac{1}{k} + 0{,}2$$
$$\omega_\eta = \frac{\omega_\varepsilon}{2}$$

Para o K-PCX que competiu, o parâmetro *k* foi definido com valor constante igual a 10, e o parâmetro *α* foi definido como uma fração das avaliações da função realizadas pelo número total de avaliações da função. Isto implica que na geração inicial, tem-se $\omega_\varepsilon = 1{,}2 - \frac{1}{k} = 1{,}1$; e à medida que a geração vai sendo processada, esse valor vai diminuindo gradativamente e ao final do processo $\omega_\varepsilon = 0{,}2$.

Após esta etapa, inicia-se o RP para escolher as *r* soluções da população, e no K-PCX abordado, essas soluções são randomicamente selecionadas dentro de toda a população. A partir de então, tem-se um reservatório de tamanho $(r + \lambda)$ constituída de *r* soluções escolhidas na população e *λ* novas soluções criadas pelo GP. A população atual é agora atualizada pelo UP, no qual as r soluções escolhidas

no RP são substituídas pelas melhores r soluções no reservatório. Essa operação assegura uma estratégia *elite-preservation* (preservação da elite).

A configuração completa feita no K-PCX para a competição no CEC 2005 foi: N = 300, k = 10, μ = 3, λ = 2 e r = 1.

3.9 LR-CMAES (ou L-CMA-ES)

O LR-CMA-ES (ou L-CMA-ES) é baseado no R-CMA-ES, que é exatamente o mesmo algoritmo em que é baseado o IPOP-CMA-ES, porém este apenas utiliza a estratégia de reinício adicionada ao CMA-ES original, sem incrementar a população a cada reinício, como ocorre no IPOP-CMA-ES.

Hansen e Auger (2013) afirmam que o CMA-ES foi originalmente desenvolvido com o intuito de melhorar a busca local das estratégias evolutivas. Ele reduz o número de avaliações da função para resolver problemas quadráticos mal escalados em várias ordens de grandeza. Porém, o algoritmo CMA-ES possui duas características singulares: quanto maior é a população, maior é o poder de busca global do CMA-ES, ao mesmo tempo que um comprimento menor do passo de cada iteração lhe concede uma característica mais de buca local. Portanto, é possível afirmar que embora o CMA-ES tenha sido inicialmente desenvolvido visando melhores resultados em buscas locais, dependendo do tamanho da população, ele pode oferecer surpreendentes bons resultados também como otimizador global.

O LR-CMA-ES que foi utilizado na competição do CEC 2005 foi propositalmente modificado de forma a ressaltar suas características de busca local, e para isso, ele teve o seu passo inicial reduzido em 100 vezes ao valor recomendado como padrão para o algoritmo CMA-ES. Além disso, o LR-CMA-ES utilizado na competição manteve o tamanho da população padrão do CMA-ES, o que equivale a algo entre 10 e 15 vezes a quantidade de dimensões do problema, ou seja, esse algoritmo foi configurado para ter desempenho de busca local muito competitivo.

O algoritmo resultante dessa configuração pode ser reconhecido como um algoritmo Buca Local avançado, por 2 motivos:

1) A matriz completa de covariância da distribuição de busca está eficientemente adaptada à topografia local da função objetivo;

2) A adaptação do tamanho do passo pode resultar em passos consideravelmente mais largos mesmo que o tamanho do passo inicial seja definido (configurado) para ser pequeno.

Ainda segundo Hansen e Auger (2013), que foram os autores dos dois algoritmos CMA-ES que competiram no CEC 2005, sempre que se forem comparar os desempenhos de algoritmos diferentes ao otimizar problemas, principalmente os multimodais, é necessário levar-se em conta que alguns algoritmos podem ter uma menor probabilidade de sucesso, porém convergem mais rapidamente em algumas áreas onde outros algoritmos tenham mais probabilidade de sucesso porém sejam mais lentos. Isto aponta nitidamente que esses autores quiseram comparar o desempenho dos seus CMA-ES da seguinte forma: descobrir quem era melhor em cada categoria de problemas, se um algoritmo com característica global (IPOP-CMA-ES) ou um algoritmo com característica local (LR-CMA-ES), para as diferentes categorias de funções.

Uma análise breve do resultado comparativo entre esses dois algoritmos apenas, mostra que seus desempenhos com funções unimodais são similares, apontando uma discreta vitória do IPOP-CMA-ES. Nas funções multimodais, o LR-CMA-ES derrotou significativamente o IPOP-CMA-ES em 4 testes de 60, enquanto foi derrotado pelo mesmo em 29 testes, e tiveram desempenhos considerados similares nos outros 27 testes, então foi uma considerável vitória do algoritmo IPOP-CMA-ES. Algumas funções não puderam ser solucionadas por nenhum dos dois, e estas foram comparadas pelo melhor valor obtido na útlima execução de cada algoritmo, e novamente houve uma vitória do algoritmo IPOP-CMA-ES.

Em todas as comparações, a diferença sempre ficava mais notavelmente vantajosa para o IPOP-CMA-ES quando se ampliavam as dimensões do problema de 10 para 30 dimensões.

Uma breve análise particular feita pelo autor deste trabalho sugere que essa vitória do IPOP-CMA-ES sobre o LR-CMA-ES muito provavelmente foi motivada principalmente pelo tamanho da população com que eles trabalham. Como ambos os algoritmos são derivados do CMA-ES puro, eles deveriam ter performances mais equilibradas, porém como o LR-CMA-ES teve seu comprimento de passo inicial muito reduzido enquanto sua população foi mantida dentro dos padrões recomendados, isto tornou o algoritmo com uma característica muito local, prejudicando sua robustez em problemas multimodais, onde ótimo global costuma

estar “mascarado” entre vários ótimos locais. Talvez, se o LR-CMA-ES tivesse sido configurado com o passo inicial menor porém com a população um pouco maior, isto iria lhe conferir uma forte característica de busca local, porém a população maior poderia lhe conceder uma boa melhora em sua característica de busca global, tornando-o mais equilibrado em comparação ao IPOP-CMA-ES.

3.10 L-SaDE

SaDE é uma sigla que significa *Self-adaptive Differential Evolution*, que traduzindo significa Evolução Diferencial Auto-adaptativa. Segundo Qin e Suganthan (2013), o algoritmo DE puro, proposto por Storn e Price, é uma poderosa técnica de busca estocástica e baseada em população, para resolver problemas de otimização, com eficiência e eficácia comprovadas e aplicadas em diversos campos de aplicação como reconhecimento de padrões, comunicação, engenharia mecânica, etc. Todavia, o DE possui parâmetros de controle e estratégias de aprendizado altamente dependentes dos problemas em consideração, o que implica dizer que para uma tarefa específica, é necessário desperdiçar uma boa quantia de tempo tentando encontrar a melhor estratégia a ser adotada, e os melhores ajustes de parâmetros a serem configurados. Este foi o motivo que levou Qin e Suganthan (2013) a desenvolver um algoritmo auto-adaptativo baseado no algoritmo DE puro e que resolvesse diversos problemas mais eficientemente, o SaDE.

No algoritmo proposto por eles, duas estratégias de aprendizado do DE são selecionadas como candidatas devido aos seus bons desempenhos em problemas com diferentes características. Essas duas estratégias de busca são escolhidas para serem aplicadas aos indivíduos na população corrente, com probabilidades proporcionais às suas prévias e bem sucedidas taxas para gerar novas soluções potencialmente boas. Além disso, 2 dos 3 parâmetros críticos associados ao DE original, C_R (constante de crossover) e F (fator para medir vetores diferenciais, contido no intervalo [0,2]), são adaptativamente modificados em vez de utilizarem valores fixos para lidar com diferentes tipos de problemas. Outro parâmetro crítico do DE, N_P (tamanho da população), permanece como uma variável especificada pelo usuário para tratar problemas com complexidade diferente.

Ainda acerca do algoritmo DE, Qin e Suganthan (2013) comentam que uma boa maneira de se inicializar a população N_P é distribuindo-se ela de maneira

uniformemente espalhada dentro dos limites do espaço de busca. Eles também citam como as cinco estratégias de mutação mais comumente usadas, as seguintes:

- “DE/rand/1”: vetor a sofrer mutação deve ser escolhido randomicamente, e apenas uma operação de diferenciação, entre dois vetores escolhidos aleatoriamente, deve ser usada no processo de mutação;
- “DE/best/1”: vetor para sofrer mutação deve ser o melhor vetor na população atual, e apenas uma operação de diferenciação, entre dois vetores escolhidos aleatoriamente, deve ser usada no processo de mutação;
- “DE/current to best/2”: vetor a sofrer mutação deve ser um vetor com índice *i*, que deve sofrer uma mutação com a diferenciação do melhor vetor da população atual em relação a ele, e em seguida uma operação de diferenciação deve ser usada, no processo de mutação, entre dois vetores escolhidos randomicamente e preferencialmente diferentes do vetor de índice *i*;
- “DE/rand/2”: vetor a sofrer mutação deve ser escolhido randomicamente, e duas operações de diferenciação, entre quatro vetores escolhidos aleatoriamente, devem ser usadas no processo de mutação;
- “DE/best/2”: vetor para sofrer mutação deve ser o melhor vetor na população atual, e duas operações de diferenciação, entre quatro vetores escolhidos aleatoriamente, devem ser usadas no processo de mutação.

A ideia principal por trás do SaDE é selecionar probabilisticamente uma das muitas estratégias de mutação disponíveis e aplica-la à população atual. Em consequência disto, têm-se muitas estratégias candidatas a serem escolhidas e também é necessário que se desenvolva um procedimento para determinar a probabilidade de se aplicar cada estratégia de mutação. Na implementação que competiu no CEC 2005, as duas estratégias escolhidas foram “rand/1/bin” e “current to best/2/bin”.

Os desenvolvedores do SaDE justificaram a escolha dessas duas estratégias devido aos seus constantes usos em muitas literaturas sobre DE e

também porque essas duas estratégias têm bons desempenhos em problemas com características distintas. A estratégia “rand/1/bin” costuma demonstrar boa diversidade populacional, enquanto que a “current to best/2/bin” exibe uma boa propriedade de convergência.

Após fazer a escolha dessas duas estratégias de mutação, Qin e Suganthan elaboraram um cálculo para se adotar uma ou outra estratégia a cada indivíduo na população. Assumindo-se que a probabilidade de se aplicar “rand/1/bin” seja *p1* e consequentemente, a probabilidade de se adotar a outra estratégia de aprendizado seja *p2*, então sabe-se que *p2 = 1 – p1*. Como inicialmente as probabilidades de se utilizar “rand/1/bin” ou “current to best/2/bin” são iguais, então inicialmente elas são configuradas como *p1 = p2 =* 0,5. Caso o j-ésimo valor de um vetor for menor ou igual a *p1*, então deve ser aplicada a estratégia “rand/1/bin” ao j-ésimo indivíduo na população atual, ao passo que se esse valor for maior que *p1*, então o j-ésimo indivíduo receberá a estratégia “current to best/2/bin”. Após a avaliação dos novos vetores experimentais, o número de vetores que obtiverem sucesso e entrarem na nova geração da população, e que foram gerados pela estratégia “rand/1/bin” deve ser armazenado em *ns1*, enquanto que o número dos que não obtiveram sucesso em compor a nova geração deve ser armazenado em *nf1*. O mesmo deve ocorrer com os vetores gerados pela estratégia “current to best/2/bin”, *ns2* e *nf2*, respectivamente. Os valores de *p1* e de *p2* são mantidos durante uma quantidade específica de gerações (50, no algoritmo que competiu), chamado “período de aprendizagem”. Então, a probabilidade de *p1* é atualizada de acordo com a seguinte fórmula:

$$p1 = \frac{ns1*(ns2+nf2)}{ns2*(ns1+nf1)+ns1*(ns2+nf2)}$$; e tem-se $p2 = 1 - p1$.

Então, após as 50 gerações, os valores de *p1* e *p2* são alterados segundo essa fórmula, e também os valores de *ns1*, *nf1*, *ns2* e *nf2* são zerados, para se evitar acúmulo de efeito colateral dos estágios de aprendizagem anteriores.

No SaDE a população N_P é mantida como um parâmetro definido pelo usuário, para que se possa lidar melhor com diferentes dimensionalidades. O parâmetro F, que é mais relacionado com a velocidade de convergência, no algoritmo DE original geralmente são usados valores para F no intervalo (0,1],

porém no SaDE proposto por Qin e Suganthan, o valor F foi pensado de modo a dar mais flexibilidade ao algoritmo, e foi estabelecido que seus valores encontram-se no intervalo (0,2], com distribuição normal de média 0,5 e desvio padrão 0,3. Dessa forma, é possível se manter boas tanto a característica local (com pequenos valores para F) quanto global (com maiores valores) para gerar bons vetores candidatos no processo evolutivo do algoritmo.

Outro parâmetro que sofreu modificação em relação ao DE original foi o C_R. Sabendo-se que desempenha um papel no DE original, o C_R deve ser cautelosamente escolhido. Sua escolha apropriada pode levar a bons desempenhos em estratégias de aprendizagem, enquanto que sua escolha inadequada pode representar uma considerável deterioração no desempenho da estratégia. Por esse motivo, no SaDE foi feita também uma modificação no comportamento de C_R, tornando-o ser distribuído como forma de uma distribuição normal de média C_{Rm} e desvio padrão igual a 0,1. Inicialmente, C_{Rm} é definido como 0,5 e são gerados diferentes valores de C_R, de acordo com essa distribuição, para os indivíduos da população. Esses valores de C_R são mantidos durante um certo número de gerações (5 no algoritmo da competição) e então um novo valor de C_R é atribuído à população sob a mesma distribuição normal. Durante cada geração, os valores de C_R associados a vetores experimentais aprovados para a próxima geração são armazenados. Após um número específico de gerações (25 no SaDE que foi para a competição), C_R é modificado várias vezes (25/5 = 5, para o algoritmo competidor) sob a mesma distribuição normal, e então nesse momento a média C_{Rm} é recalculada de acordo com os valores gravados de C_R correspondentes aos vetores bem-sucedidos durante o período de análise. O desvio padrão é mantido como 0,1. Desta forma, o C_R consegue aprender com o comportamento do algoritmo e adaptar-se ao problema em questão. Também deve ser destacado que os valores de C_R dos vetores experimentais bem-sucedidos são zerados após recalcular a nova média, para evitar o possível acúmulo de efeitos inapropriados.

Para acelerar ainda mais a convergência do SaDE, foi aplicado um procedimento de Busca Local (LS) após um certo período, que no algoritmo utilizado no CEC 2005 equivale a 200 gerações. Esta técnica é aplicada apenas a 5% dos indivíduos, incluindo os melhores indivíduos encontrados mais distantes e os indivíduos selecionados aleatoriamente fora dos 50% melhores indivíduos da

população atual. Nesta busca local é empregado o método Quasi-Newton. Um operador de busca local é necessário porque o valor máximo de execuções pré-determinado é muito pequeno para atingir o nível de precisão desejado. Após essa inserção de um LS ao SaDE, o algoritmo passa a ser chamado de L-SaDE.

O L-SaDE foi configurado para a competição com os valores de parâmetros já citados neste item do trabalho, e o tamanho da sua população foi configurado como 50 e 100 indivíduos, para os problemas com 10-D e 30-D, respectivamente. Esse algoritmo trabalho bem com quase todas as funções de 1 a 15, tanto em 10-D quanto em 30-D, porém nas funções de 16-25, esse algoritmo falhou e não conseguiu obter o ótimo global para nenhuma delas. Segundo Qin e Suganthan, isto foi ocasionado pela elevada multimodalidade dessas funções compostas, e também o LS embutido nele fez com que o algoritmo convergisse muito rapidamente para ótimos locais, falhando em encontrar o ótimo global.

Comparando o L-SaDE com o DE puro, é possível notar que em 10-D o DE mostrou resultados melhores. Isto é ocasionado porque o L-SaDE possui um algoritmo mais complexo e que exige um pouco mais de execuções para encontrar o ótimo global, porém ao se trabalhar com 30-D, o L-SaDE supera facilmente o DE pelo mesmo motivo, ou seja, como seu algoritmo é mais complexo, mesmo ele exigindo mais execuções para problemas simples, ele consegue lidar melhor quando se aumenta o nível de complexidade do problema.

3.11 SPC-PNX

O algoritmo SPC-PNX foi desenvolvido por Pedro J. Ballester, John Stephenson, Jonathan N. Carter e Kerry Gallagher (2013), e seu nome é uma sigla dividida em duas partes, onde SPC representa o operador de substituição utilizado e PNX representa o operador de cruzamento adotado nesse algoritmo. O SPC-PNX é um algoritmo genético de parâmetro real, do tipo *steady-state*.

Gallagher et al. (2013) explica que no SPC-PNX, 2 genitores são selecionados da população de tamanho N para produzir λ descendentes através do operador de cruzamento. Após isso, o valor da função objetivo associado a cada descendente é avaliado. A prole (descendência) e a população atual são então combinadas, de modo que a população permaneça com um tamanho constante através do operador de substituição. Essas 4 operações (seleção, cruzamento,

avaliação de *fitness* e substituição) caracterizam, em algoritmos genéticos, uma geração, e no SPC-PNX isto não é diferente.

No algoritmo abordado neste item 3.11, a seleção é feita de forma randômica e uniforme, e são selecionados 2 membros da população atual que serão os genitores. Diferentemente da maioria dos algoritmos genéticos existentes, no SPC-PNX o valor de *fitness* não é levado em consideração durante este processo, ou seja, a seleção é realmente aleatória. Os desenvolvedores deste algoritmo justificam essa seleção alegando que a mesma permite explorar de forma mais intensa a informação contida em diversas soluções com baixo *fitness*.

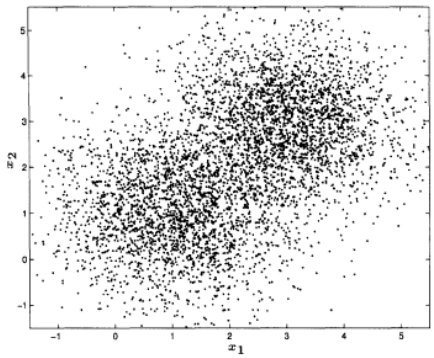
Este algoritmo, como seu nome sugere, na etapa de cruzamento utiliza o operador PNX. Este é um operador *parent-centric*, auto-adaptativo no sentido de que a propagação das possíveis soluções dos descendentes depende da distância entre os genitores, a qual diminui à medida que a população converge. O PNX também é isotrópico já que ele não realiza a busca preferencialmente ao longo de qualquer direção particular. Outra vantagem é que o PNX possui uma probabilidade diferente de zero, de gerar descendentes por todo o espaço de busca. Essas características contribuem para uma exploração mais ampla do espaço de busca.

No PNX, o j-ésimo gene (y_j) é determinado para cada membro de λ, da seguinte forma: primeiramente escolhe-se aleatoriamente um número $\omega \in [0,1]$, e é usada a forma $y_j^{(1)}$ se $\omega < 0{,}5$ ou $y_j^{(2)}$ caso $\omega \geq 0{,}5$. Uma vez que essa escolha é feita, a mesma forma selecionada é usada para cada componente *j*. As formas são:

$$y_j^{(1)} = N(x_j^{(1)}, |x_j^{(2)} - x_j^{(1)}|/\eta)$$
$$y_j^{(2)} = N(x_j^{(2)}, |x_j^{(2)} - x_j^{(1)}|/\eta)$$

onde $N(\mu, \sigma)$ é um número aleatório obtido de uma distribuição Gaussiana com média μ e desvio padrão σ, e $x_j^{(i)}$ é o j-ésimo componente do i-ésimo genitor e η é um parâmetro ajustável que, quanto maior o seu valor, mais concentrada é a busca ao redor dos genitores, como demonstrado na figura 07, na qual é demonstrada uma prole gerada a partir dos genitores $x^{(1)} = (1,1)$ e $x^{(2)} = (3,3)$, para um PNX com $\eta = 2$:

Figura 07 – Prole gerada por um operador PNX



Fonte: Gallagher et al. (2013, p. 499)

Para o processo de substituição, o SPC-PNX faz uso do SPCS (*scaled probabilistic crowding scheme*). No SPCS, para cada descendência (prole) são escolhidos randomicamente N_{REP} indivíduos da população atual. Esses indivíduos irão competir com a prole por um lugar na população de acordo com as seguintes probabilidades de escolha:

$$p(\vec{x}^{ofp}) = \frac{f(\vec{x}^{ofp}) - f_{best}}{f(\vec{x}^{ofp}) + f(\vec{x}^{cst}) - 2f_{best}}$$
$$p(\vec{x}^{cst}) = \frac{f(\vec{x}^{cst}) - f_{best}}{f(\vec{x}^{ofp}) + f(\vec{x}^{cst}) - 2f_{best}}$$

onde $f(\vec{x})$ é o valor da função objetivo para um indivíduo $\vec{x}$, e f_{best} é o valor da função para o melhor indivíduo entre os da prole e do grupo de N_{REP} indivíduos.

Gallagher et al. (2013) afirma ainda que esse esquema SPCS traz muitos benefícios ao SPC-PNX, como por exemplo, o fato de que o indivíduo com melhor *fitness* nem sempre vence, e isso ajuda a prevenir convergência prematura. Os esquemas de agrupamento como esse promovem a criação de subpopulações que exploram diferentes regiões do espaço de busca.

Para a competição no CEC 2005, este algoritmo teve seus parâmetros de controle configurados da seguinte maneira: $\eta = 2$, $\lambda = 1$ e $N_{REP} = 2$.

Além desses parâmetros, deve-se lembrar que o parâmetro *N* foi definido independentemente para cada problema, pelo PNX descrito anteriormente. Deve-se destacar também que isso foi feito para simplicar o ajuste do algoritmo, embora os

desenvolvedores dele acreditassem que ajustar os parâmetros de controle podem implicar em melhoras consideráveis de desempenho.

O quadro 08 a seguir mostra os valores adotados para N em cada função, para 10 e para 30 dimensões:

Quadro 08 – Valores de *N* adotados por SPC-PNX para cada função

D=10	1	2	3	4	5	6	7	8	9	10	11	12
N	20	40	45	40	35	60	150	1000	225	200	100	200
D=30	1	2	3	4	5	6	7	8	9	10	11	12
N	40	40	45	40	100	60	150	1000	300	200	200	200

D=10	13	14	15	16	17	18	19	20	21	22	23	24	25
N	50	100	150	150	150	100	100	100	50	400	400	150	200
D=30	13	14	15	16	17	18	19	20	21	22	23	24	25
N	55	100	150	150	150	300	300	300	300	300	300	150	300

Fonte: Gallagher (2013, p. 501)

Ainda acerca do SPC-PNX, deve ser dito que os seus desenvolvedores resolveram criar uma versão constrita do algoritmo para concentrar a busca dentro da região de inicialização, que força o algoritmo a concentrar a busca dentro da região de inicialização. Isso foi feito inserindo uma pequena modificação no operador de cruzamento PNX: só aceitar o descendente criado y_j se e somente se ele estiver localizado dentro dos limites da região de inicialização, do contrário, usar o componente x_j do genitor correspondente no lugar dele, ou seja, $y_j = x_j$. A versão natural do algoritmo, ou seja, a não-constrita, foi aplicada apenas aos problemas de número 7 e 25, que possuem o ótimo global fora do espaço de inicialização. Para todas as outras funções foi usada a versão constrita do SPC-PCX.

4 COMPETIÇÃO

Reiterando o que foi dito nos capítulos anteriores, o IEEE em parceria com a *Evolutionary Programming Society*, resolveu elaborar um conjunto de testes destinados a avaliar o desempenho de diversos algoritmos ligados à computação evolutiva. Esse conjunto de testes foi reunido em um evento, e este útltimo foi criado no ano de 1999, ocorre uma vez a cada ano e seu nome é CEC, que significa *Congress on Evoltuionary Computation*.

O CEC é um evento que aborda diversos temas ligados à computação evolutiva, que vão desde simples publicações a respeito desta área, até competições entre robôs com inteligência artificial. Dentre esses diversos temas abordados pelo CEC, um que chamou muito a atenção de todos os profissionais da área da computação evolutiva foi o de Otimização de Parâmetros Reais. Este é um campo da computação que está sempre em foco, pois seus experimentos e resultados obtidos têm aplicações em diversos campos não só da computação, mas de diversas áreas profissionais, como por exemplo, as telecomunicações, as ciências espaciais, diversas engenharias, robótica, etc.

Dentre os CECs que já aconteceram, um deles merece ser destacado para a computação evolutiva: o CEC 2005. Neste, foram reunidas 25 funções com o intuito de desafiar qualquer algoritmo a resolvê-las, e seriam medidos para cada algoritmo, a capacidade do mesmo em resolver as funções. O principal critério é encontrar o ótimo global da função, porém, como são funções que trabalham com parâmetros reais, isto implica em um caso de otimização contínua, ou seja, tem-se um espaço de busca baseado no conjunto dos números reais. Desta forma, é bem mais complicado medir-se com exatidão o resultado ótimo, preciso, perfeito, e como já mencionado na introdução, não é este o objetivo das meta-heurísticas, e sim encontrar um resultado ótimo que seja muito bom, e o mais próximo possível do ótimo global exato.

No item 4.1 que segue, são mostradas as 25 funções que serviram de desafio para os algoritmos competidores, e no item 4.2 são mostrados os resultados do teste, bem como os desempenhos dos algoritmos e um comparativo dos seus desempenhos de forma geral.

4.1 METODOLOGIA, FUNÇÕES E MÉTRICAS

Utilizando os relatos dos estudos desenvolvidos por Suganthan ET AL (2013), apresentam-se a seguir as definições de problemas e critérios de avaliação das 25 funções-teste para o CEC 2005. O autor utiliza dois tipos de funções: as unimodais e a multimodais.

4.1.1 Funções unimodais

As funções unimodais utilizadas no CEC 2005 são as seguintes:

- *Shifted Sphere Function (F1);*
- *Shifted Schwefel's Problem 1.2 (F2);*
- *Shifted Rotated High Conditioned Elliptic Function(F3);*
- *Shifted Schwefel's Problem 1.2 with Noise in Fitness (F4);*
- *Schwefel's Problem 2.6 with Global Optimum on Bounds (F5).*

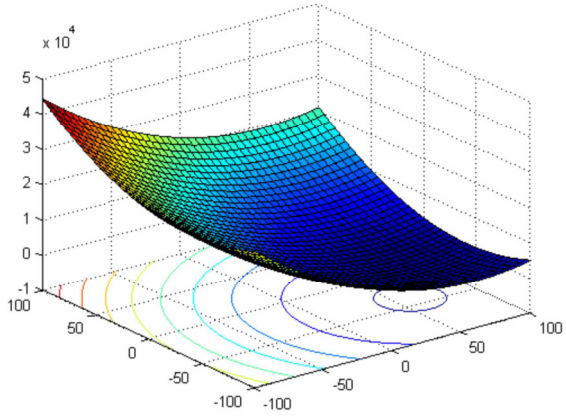
A seguir, tem-se uma descrição mais complete e detalhada sobre cada uma delas.

F1 - S*hifted Sphere Function*

$F_1(x) = \sum_{i=1}^{D} z_i^2 + f\_bias_1$, z = x – o, x = $[x_1, x_2, \dots, x_D]$

D: dimensões. o = [$o_1, o_2, \dots, o_D$] : O ótimo global deslocado

Figura 08 – Mapa 3-D para função F1 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Unimodal
- Deslocada
- Separável
- Escalável
- X ϵ [-100, 100]^D, Ótimo global: $x^* = o, F_1(x^*) = f\_bias_l$ = - 450

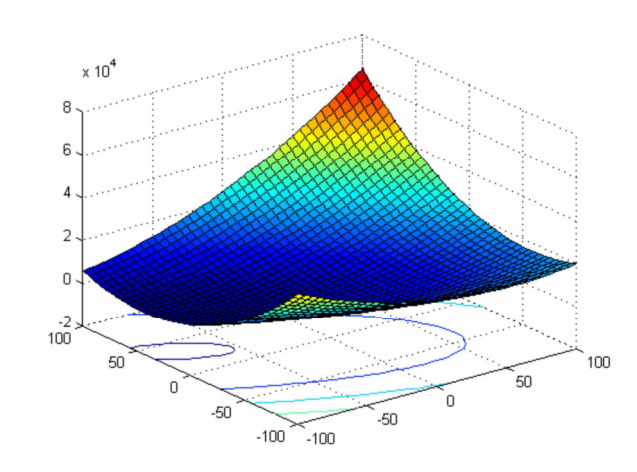
F2 - *Shifted Schwefel's Problem 1.2*

F_2(x) = $\sum_{i=1}^{D}$ $(\sum_{j=1}^{i} z_j)^2 + f\_bias_2$, z = x – o, x = [$x_1, x_2, \dots, x_D$]

D: dimensões

o = [$o_1, o_2, \dots, o_D$] : O ótimo global deslocado

Figura 09 – Mapa 3-D para função F2 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Unimodal
- Deslocada
- Não separável
- Escalável
- X ϵ [-100, 100]^D, Ótimo global: $x^* = o, F_2(x^*) = f\_bias_2$ = - 450

F3 - *Shifted Rotated High Conditioned Elliptic Function*

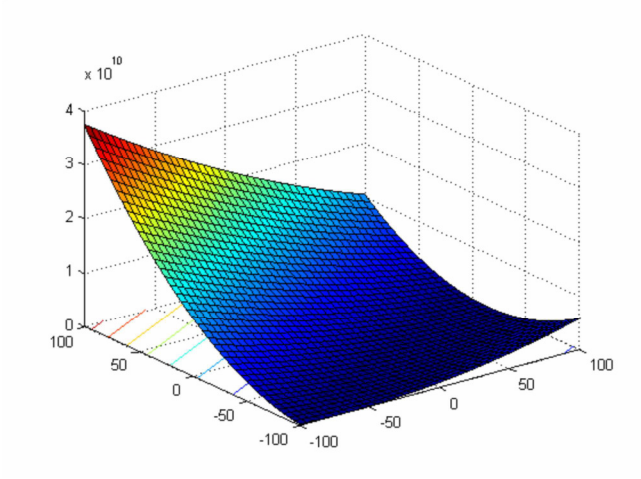
F_3(x) = $\sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}} z_1^2 + f\_bias_1$, z = (x – o) * M, x = [$x_1, x_2, \dots, x_D$]

D: dimensões

o = [$o_1, o_2, \dots, o_D$] : O ótimo global deslocado

M: Matriz ortogonal

Figura 10 – Mapa 3-D para função F3 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Unimodal
- Deslocada
- Rotacionada
- Não separável
- X ϵ [-100, 100]^D, Ótimo global: $x^* = o, F_3(x^*) = f\_bias_3$ = - 450

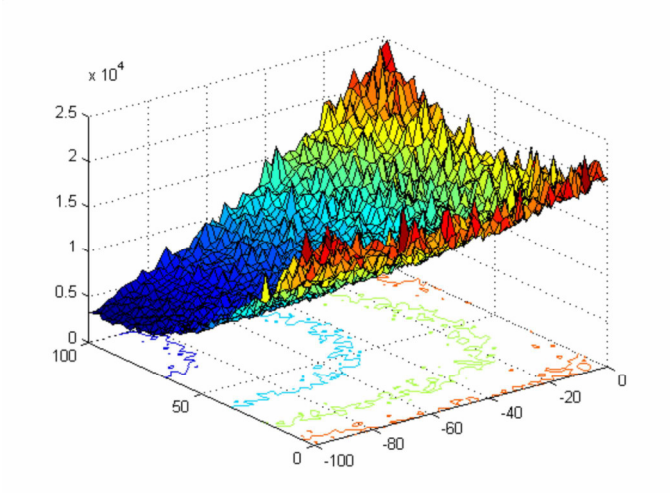
F4 - *Shifted Schwefel's Problem 1.2 with Noise in Fitness*

$$F_4(x) = (\sum_{i=1}^{D}(\sum_{j=1}^{i} z_j)^2) * (1 + 0.4|N(0,1)|) + f\_bias_4, z = x - o, x = [x_1, x_2, \dots, x_D]$$

D: dimensões

o = [$o_1, o_2, \dots, o_D$] : O ótimo global deslocado

Figura 11 – Mapa 3-D para função F4 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Unimodal
- Deslocada
- Não separável
- Escalável
- Ruídos em fitness
- X ϵ [-100, 100]^D, Ótimo global: $\boldsymbol{x}^* = o, F_4(\boldsymbol{x}^*) = f\_bias_4 =$ - 450

F5 - *Schwefel's Problem 2.6 with Global Optimum on Bounds*

f (**x**) = max { $|x_1 + 2x_2 - 7|$, | 2 x_1 +x_2 - 5| }, *i* = 1, …, *n*, $\boldsymbol{x}^*$ = [1,3], *f* (x^*) = 0

Estender para D dimensões:

F_5(**x**) max{ | $\boldsymbol{A}_i\boldsymbol{x} - \boldsymbol{B}_i$| } + $f\_bias_5, i = 1, \dots, D, \boldsymbol{x} = [x_1, x_2, \dots, x_D]$

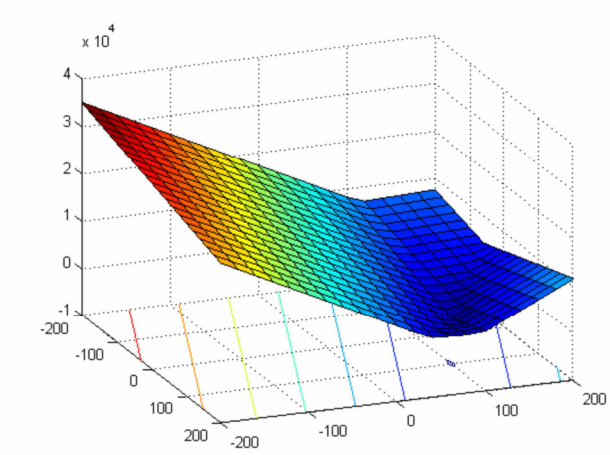
D: dimensões

A é uma matriz *D*D*, a_{ij} são números aleatórios inteiros no intervalo [*-500, 500*], det(**A**) ≠ 0 ,A_i é a i^{th} da linha de **A**.

$\boldsymbol{B}_i = \boldsymbol{A}_i * \mathbf{o}$, **o** é o vetor *D**1, o_i um número aleatório no intervalo [-100, 100]

Depois de carregar o arquivo de dados, configurar o_i= -100, para *i = 1,2,...,*$\lceil D/4 \rceil$, o_i= 100, para *i* = $\lfloor 3D/4 \rfloor$*,...., D*

Figura 12 – Mapa 3-D para função F5 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Unimodal
- Não separável
- Escalável
- Se o procedimento de inicialização inicializa a população nos limites, este problema será resolvido facilmente.
- **x** ϵ [-100, 100]^D, Ótimo global: $\boldsymbol{x}^* = o, F_5(\boldsymbol{x}^*) = f\_bias_5 =$ - 310

4.1.2 Funções multimodais

As funções multimodais utilizadas no CEC 2005, apresentadas e descritas a seguir são divididas em três categorias: funções básicas, funções expandidas e funções de composição híbrida. Compõem as **funções básicas**:

- *Shifted Rosenbrock's Function (F6);*
- *Shifted Rotated Griewank's Function without Bounds (F7);*
- *Shifted Rotated Ackley's Function with Global Optimum on Bounds (F8);*
- *Shifted Rastrigin's Function (F9);*
- *Shifted Rotated Rastrigin's Function (F10);*
- *Shifted Rotated Weierstrass Function (F11);*
- *Schwefel's Problem 2.13 (F12).*

As **funções expandidas** são:

- Expanded Extended Griewank's plus Rosenbrock's Function [F8F2] (F13);
- Shifted Rotated Expanded Scaffer's F6 *(F14).*

*Já as **funções de composição híbrida** são:*

- *Hybrid Composition Function (F15);*
- *Rotated Hybrid Composition Function (F16);*
- *Rotated Hybrid Composition Function with Noise in Fitness (F17);*
- *Rotated Hybrid Composition Function (F18);*
- *Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum F(19);*
- *Rotated Hybrid Composition Function with the Global Optimum on the Bounds (F20);*
- *Rotated Hybrid Composition Function (F21);*
- *Rotated Hybrid Composition Function with High Condition Number Matrix (F22);*
- *Non-Continuous Rotated Hybrid Composition Function (F23);*
- *Rotated Hybrid Composition Function (F24);*
- *Rotated Hybrid Composition Function without Bounds (F25).*

4.1.2.1 Funções multimodais básicas

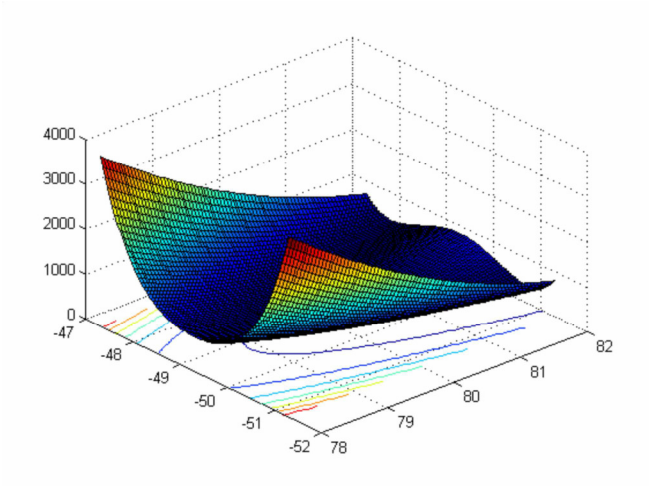
F6 - *Shifted Rosenbrock's Function*

F_6 (x) = $\sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f\_bias_6, \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} + 1, \boldsymbol{x} = [x_1, x_2, \dots, x_D]$

D: Dimensões

o = [$o_1, o_2, \dots, o_D$] : O ótimo global deslocado

Figura 13 – Mapa 3-D para função F6 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Deslocada
- Não separável
- Escalável
- Possui um vale muito estreito do ótimo local para o ótimo global
- x ϵ [-100, 100]^D, Ótimo global: $\boldsymbol{x}^* = o, F_6(\boldsymbol{x}^*) = f\_bias_6 = 390$

F7 - Shifted Rotated Griewank's Function without Bounds

F_7 (x) = $\sum_{i=1}^{D} \frac{Z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_1}{\sqrt{i}}) + 1 + f\_bias_7$, **z** = (**x** - **o**) * **M**, **x** = $[x_1, x_2, \dots, x_D]$

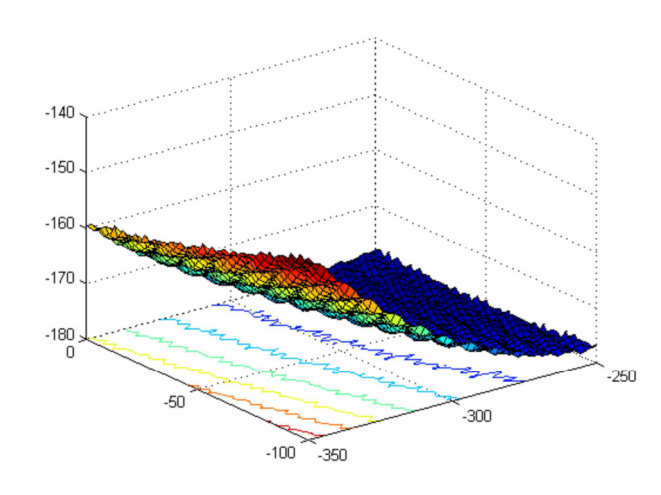
D: dimensões

o = [$o_1, o_2, \dots, o_D$] : O ótimo global deslocado

M': Matriz linear de transformação, condição número = 3

M = **M'** (1+0.3|N(0,1)|)

Figura 14 – Mapa 3-D para função F7 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Rotacionada
- Deslocada
- Não separável
- Escalável
- Não há limites para as variáveis x
- Inicializar população em D [0, 600], Ótimo global: $\boldsymbol{x}^* = o$ está fora da inicialização de alcance, $F_7(\boldsymbol{x}^*) = f\_bias_7$ = -180

F8 - *Shifted Rotated Ackley's Function with Global Optimum on Bounds*

F_8 (x) = - 20 exp (- 0.2 $\sqrt{\frac{1}{D}\sum_{I=1}^{D} z_i^2}$) - exp ($\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)$) + 20 + e + f_bias_8,

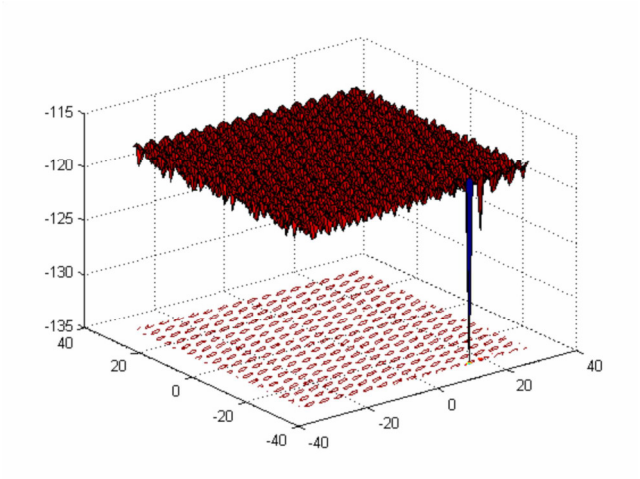
z = (**x** - **o**) * **M**, **x** = $[x_1, x_2, \ldots, x_D]$, *D: dimensões*

o = [$o_1, o_2, \ldots, o_D$] : O ótimo global deslocado

Após carregar o arquivo de dados, configurar o_{2j-1} = -32 o_{2j} são distribuídos aleatoriamente no intervalo de pesquisa, para *j* = 1,2, ..., $\lfloor D/2 \rfloor$

M = Matriz linear de transformação, condição número = 100

Figura 15 – Mapa 3-D para função F8 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Rotacionada
- Deslocada
- Não separável
- Escalável
- Número de condições A - cond (A) - aumenta com o número de variáveis como O (D2)
- Ótimo global no limite
- Se o procedimento de inicialização inicializa a população aos limites, este problema será resolvido facilmente
- x ϵ [-32, 32]^D, Ótimo global: $\boldsymbol{x}^* = o, F_8(\boldsymbol{x}^*) = f\_bias_8$ = -140

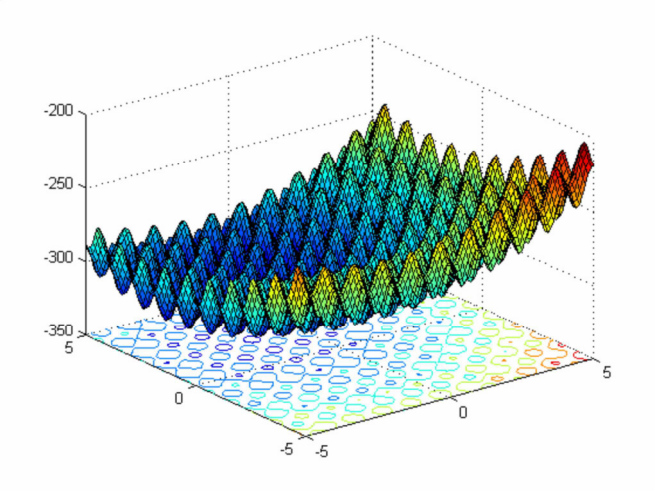
F9 - *Shifted Rastrigin's Function*

F_9 (x) = $\sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_9, \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}, \boldsymbol{x} = [x_1, x_2, \ldots, x_D]$

D: dimensões

o = [$o_1, o_2, \ldots, o_D$] : O ótimo global deslocado

Figura 16 – Mapa 3-D para função F9 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Deslocada
- Separável
- Escalável
- Quantidade de ótimos locais é grande
- x ϵ [-5, 5]^D, Ótimo global: $\boldsymbol{x}^* = o, F_9(\boldsymbol{x}^*) = f\_bias_9 =$ - 330

F10 - *Shifted Rotated Rastrigin's Function*

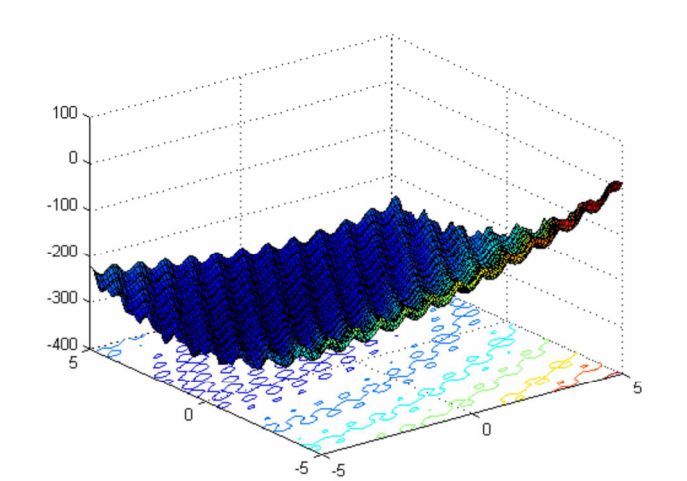
F_{10} (x) = $\sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_{10}, \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o}) * \boldsymbol{M}, \boldsymbol{x} = [x_1, x_2, \dots, x_D]$

D: dimensões

o = [$o_1, o_2, \dots, o_D$] : O ótimo global deslocado

M = Matriz linear de transformação, condição número = 2

Figura 17 – Mapa 3-D para função F10 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Deslocada
- Rotacionada
- Não-separável
- Escalável
- Quantidade de ótimos locais é grande
- x ϵ [-5, 5]^D, Ótimo global: $\boldsymbol{x}^* = o, F_{10}(\boldsymbol{x}^*) = f\_bias_{10}$ = - 330

F11 - *Shifted Rotated Weierstrass Function*

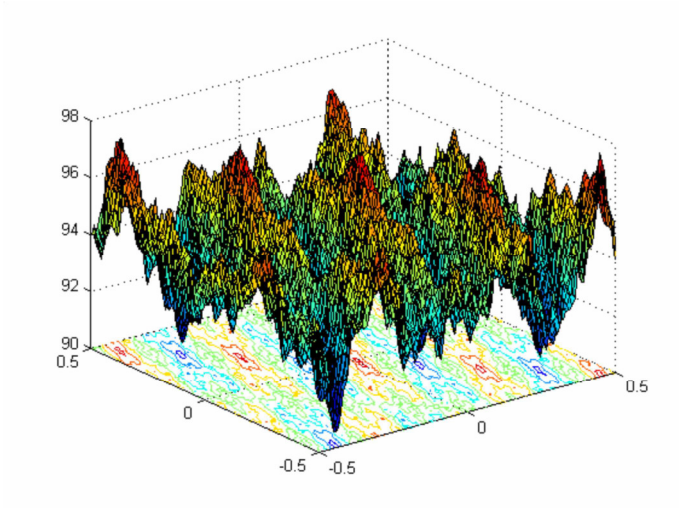
F_{11} (x) = $\sum_{i=1}^{D}(\sum_{k=0}^{kmax}[\, a^k \cos(2\pi b^k(z_i + 0.5))]) - D\sum_{k=0}^{kmax}[\, a^k \cos(2\pi b^k .0.5)] +$ f_bias_{11} , a = 0.5, b = 3, k_{max}= 20, $\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o}) * \boldsymbol{M}, \boldsymbol{x} = [x_1, x_2, \ldots, x_D]$

D: dimensões

o = [$o_1, o_2, \ldots, o_D$] : O ótimo global deslocado

M = Matriz linear de transformação, condição número = 5

Figura 18 – Mapa 3-D para função F11 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Deslocada
- Rotacionada
- Não-separável
- Escalável
- Contínuo mas diferenciável apenas num conjunto de pontos
- x ϵ [-0.5,0.5]^D, Ótimo global: $\boldsymbol{x}^* = o, F_{11}(\boldsymbol{x}^*) = f\_bias_{11} = 90$

F12 - *Schwefel's Problem 2.13*

F_{12} (x) = $\sum_{i=1}^{D}(A_i - B_i(x))^2 + f\_bias_{12}$, $x = [x_1, x_{12}, \dots, x_D]$

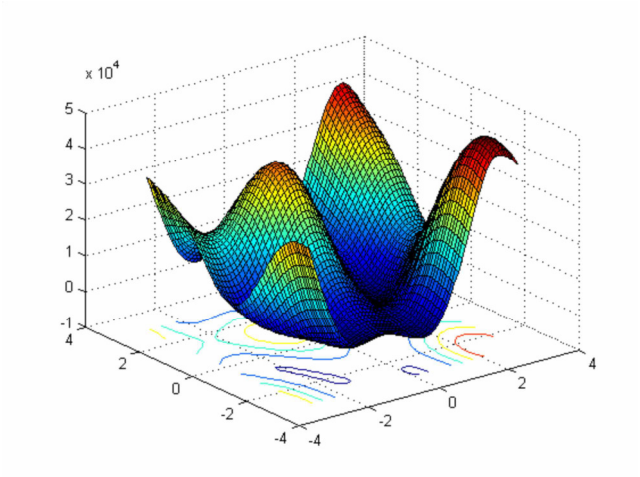
$A_1 = \sum_{j=1}^{D}(a_{ij} sin\alpha_j + b_{ij} cos\alpha_j)$, $B_i(x) = \sum_{j=1}^{D}(a_{ij} sin\, x_j + b_{ij} cos x_j)$, para i = 1,...,D

D: dimensões

A, **B** são matrizes *D*D*, a_{ij}, b_{ij} são números aleatórios inteiros no intervalo de [-100, 100],

$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_D]$, α_j são números aleatórios no intervalo [- π, π].

Figura 19 – Mapa 3-D para função F12 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Deslocada
- Não-separável
- Escalável
- x ϵ [-π,π]^D, Ótimo global: $\boldsymbol{x}^* = \boldsymbol{\alpha}, F_{12}(\boldsymbol{x}^*) = f\_bias_{12}$ = - 460

4.1.2.2 Funções multimodais expandidas

Utilizando uma função F (x, y) 2-D como função de partida, a função expandida correspondente é:

EF $(x_1, x_2, ..., x_D) = F(x_1, x_2)$ + F (x_2, x_3) + ...+ F (x_{D-1}, x_D) + F (x_D, x_1)

F13 - *Expanded Extended Griewank's plus Rosenbrock's Function (F8F2)*

F8: Griewank's Function: *F8(x)* = $\sum_{i=1}^{D} \frac{x_i^2}{4000}$ - $\prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}})$ + 1

F2: Rosenbrock's Function: *F2(x)* = $\sum_{i=1}^{D-1}(100\,(x_i^2 - x_{i+1})^2 + (x_1 - 1)^2)$

F8F2 $(x_1, x_2, ..., x_D) = F8\,(F2\,(x_1, x_2)) + F8(F2\,(x_2, x_3))+... +F8\,(F2(x_{D-1}, x_D))+ F8(F2(x_D, x_1))$

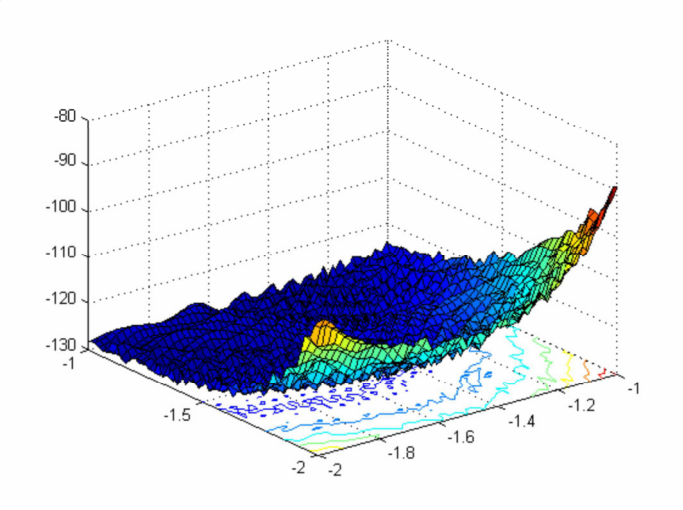
mudar para

F_{13} (x) = *F8 (F2*$((z_1, z_2))$+ *F8 (F2*$((z_2, z_3))$+... + *F8 (F2* $(z_D, z_1))$+ f_bias_{13}

Z = x – o + 1, x = $[x_1, x_2, ..., x_D]$

D: dimensões o = [$o_1, o_2, \ldots, o_D$] : O ótimo global deslocado

Figura 20 – Mapa 3-D para função F13 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Deslocada
- Não-separável
- Escalável
- x ϵ [-5,5]^D, Ótimo global: $\boldsymbol{x}^* = \boldsymbol{\alpha}, F_{13}(\boldsymbol{x}^*) = f\_bias_{13}(13) =$ - 130

F14 - *Shifted Rotated Expanded Scaffer's F6*

$F(x, y) = 0.5 + \frac{(sin^2\left(\sqrt{x^2+y^2}\right)-0.5)}{(1+0.001\ (x^2+y^2))^2}$

Expandida para

F_{14} (x) = EF (z_1, z_2 , ..., z_D) = F (z_1, z_2)+ F (z_2, z_3) + ... + F(z_D, z_1)+ f_bias_{14},

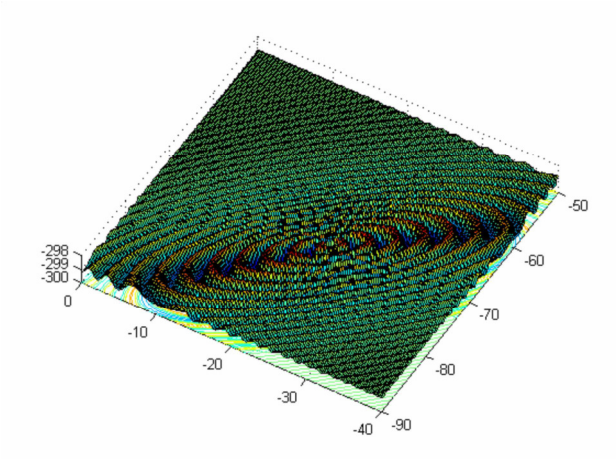
$\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o}) * \boldsymbol{M}, \boldsymbol{x} = [x_1, x_2, \ldots, x_D]$

D: dimensões

o = [$o_1, o_2, \ldots, o_D$] : O ótimo global deslocado

M = Matriz linear de transformação, condição número = 3

Figura 21 – Mapa 3-D para função F14 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Deslocada
- Não-separável
- Escalável
- x ϵ [-100,100]^D, Ótimo global: $\boldsymbol{x}^* = \boldsymbol{o}$, $F_{14}(\boldsymbol{x}^*) = f\_bias_{14}(14) =$ - 300

4.1.2.3 Funções multimodais de composição híbrida

F15 - *Hybrid Composition Function*

$f_{1-2}(x)$: *Rastrigin's Function*

$$f_i(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$$

$f_{3-4}(x)$: *Weierstrass Function*

$$f_i(x) = \sum_{i=1}^{D}(\sum_{k=0}^{kmax}[\, a^k \cos(2\pi b^k (x_i + 0.5))]) - D\sum_{k=0}^{kmax}[\, a^k \cos(2\pi b^k \cdot 0.5)]$$

$$a = 0.5, b = 3, k_{max} = 20$$

$f_{5-6}(x)$: *Griewank's Function*

$$f_i(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$$

$f_{7-8}(x)$: *Ackley's Function*

$$f_i(x) = -20\ exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$$

$f_{9-10}(x)$: *Sphere Function*

$f_i\ (x) = \sum_{i=1}^{D} x_i^2$

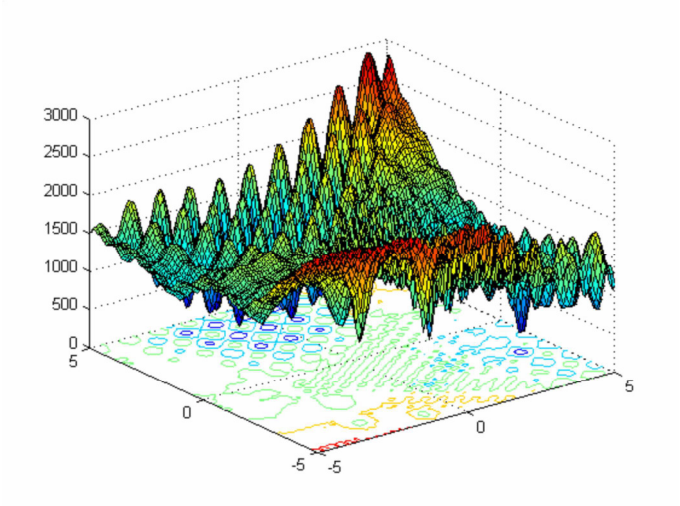
σ_i = 1 para *i* = 1,2, ..., *D*

λ = [1,1,10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100]

M_i = são todas as matrizes de identidade

Deve-se notar que estas fórmulas são apenas para as funções básicas, o deslocamento ou rotação não estão incluídos nestas expressões. X aqui é apenas uma variável em uma função. Pegue f_1 como um exemplo, quando nós calculamos f_1 (((x - o_1)/λ_1) $*$ M_1, nós precisamos calcular $f_1\ (z) = \sum_{i=1}^{D}(z_1^2 - 10\cos(2\pi z_i) + 10)$, z = ((x - o_1)/λ_1) $*$ M_1

Figura 22 – Mapa 3-D para função F15 2-D



Fonte: Suganthan ET AL (2013)

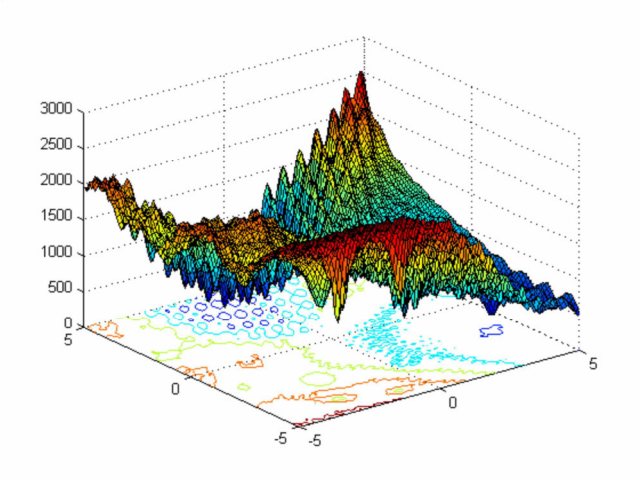
Propriedades:

- Multimodal
- Separável perto do ótimo global (Rastrigin)
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Sphere Functions entregam duas áreas planas para a função
- x ϵ $[-5,5]^D$, Ótimo global: $\boldsymbol{x}^* = \boldsymbol{o}$, $F_{15}(\boldsymbol{x}^*) = f\_bias_{15} = 120$

F16 - *Rotated Hybrid Composition Function*

Exceto $\mathbf{M}_i$ que são diferentes matrizes de transformação linear com numero de condição igual a 2, todos os outros parâmetros são os mesmos de *F*15

Figura 23 – Mapa 3-D para função F16 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Rotacionada
- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Sphere Functions entregam duas áreas planas para a função
- x ϵ [-5,5]^D, Ótimo global: $\boldsymbol{x}^* = \boldsymbol{o}$, $F_{16}(\boldsymbol{x}^*) = f\_bias_{16} = 120$

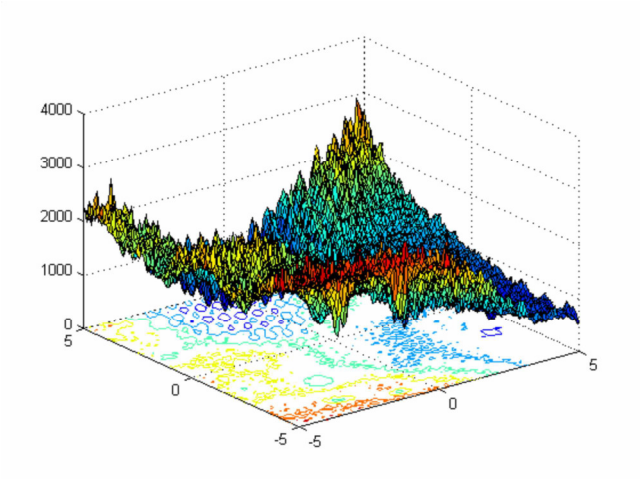
F17 - *Rotated Hybrid Composition Function with Noise in Fitness*

Deixe $(F_{16} - f\_bias_{16})\ ser\ G(x)$, então

$F_{16}(x) = G(x) * (1 + 0.2|\mathrm{N}(0,1)|) + f\_bias_{17}$

Todos os parametros são os mesmos de F16.

Figura 24 – Mapa 3-D para função F17 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Rotacionada
- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Sphere Functions entregam duas áreas planas para a função
- Com ruído Gaussiano em fitness
- x ϵ [-5,5]^D, Ótimo global: $\boldsymbol{x}^* = \boldsymbol{o}$, $F_{16}(\boldsymbol{x}^*) = f\_bias_{16} = 120$

F18 - *Rotated Hybrid Composition Function*

$f_{1-2}(x)$: *Ackley's Function*

$$f_i(x) = -20\ exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_1^2}) - exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$$

$f_{3-4}(x)$: *Rastrigin's Function*

$$f_i(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$$

$f_{5-6}(x)$: *Sphere Function*

$$f_i(x) = \sum_{i=1}^{D} x_1^2$$

$f_{7-8}(x)$: *Weierstrass Function*

$$f_i(x) = \sum_{i=1}^{D}(\sum_{k=0}^{kmax}[a^k\ \cos\left(2\pi b^k(x_i + 0.5)\right)]) - D\sum_{k=0}^{kmax}[a^k\cos(2\pi b^k \cdot 0.5)],$$

a=0.5, b=3, k_{max}=20

$f_{9-10}(x)$: *Griewank's Function*

$$f_i(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

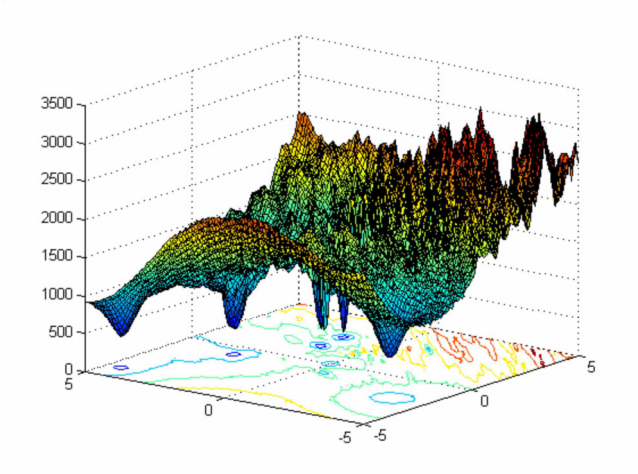
σ = [1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2];

λ = [2*5/32; 5/32; 2*1; 1; 2*5/100; 5/100; 2*10; 10; 2*5/60; 5/60]

$\boldsymbol{M}_i$ são todas as matrizes de rotação. Números de condição são [2 3 2 3 2 3 20 30 200 300]

$\mathbf{o}_{10}$= [0,0,...,0]

Figura 25 – Mapa 3-D para função F18 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Rotacionada
- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Sphere Functions entregam duas áreas planas para a função
- Um ótimo local está situado na origem
- x ϵ $[-5,5]^D$, Ótimo global: $\boldsymbol{x}^* = \mathbf{o}_1$, $F_{18}(\boldsymbol{x}^*) = f\_bias_{18} = 10$

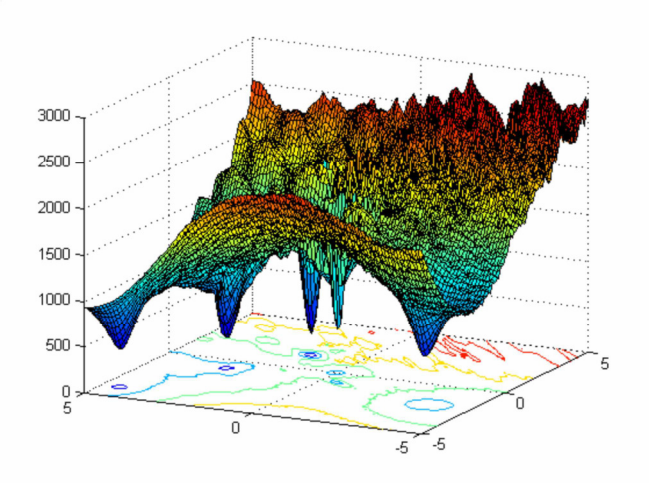
F19 - *Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum*

Todos os parametros são os mesmos de *F*18 exceto

σ = [0.1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2];,

λ = [0.1*5/32; 5/32; 2*1; 1; 2*5/100; 5/100; 2*10; 10; 2*5/60; 5/60]

Figura 26 – Mapa 3-D para função F19 2-D



Fonte: Suganthan ET AL (2013)

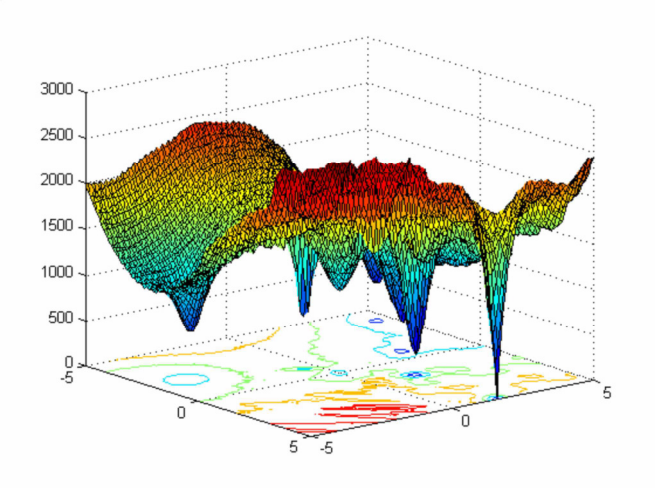
Propriedades:

- Multimodal
- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Sphere Functions entregam duas áreas planas para a função
- Um ótimo local está situado na origem
- Uma depressão estreita para um ótimo global
- x ϵ $[-5,5]^D$, Ótimo global: $\boldsymbol{x}^* = \mathbf{o}_1$, $F_{19}(\boldsymbol{x}^*) = f\_bias_{19}(19) = 10$

F20 - *Rotated Hybrid Composition Function with the Global Optimum on the Bounds*

Todas as configurações são as mesmas, exceto F18 após o carregamento do arquivo de dados, configure $o_{1(2j)}$ = 5, para *j* = 1, 2, ..., $\lfloor D/2 \rfloor$

Figura 27 – Mapa 3-D para função F20 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Sphere Functions entregam duas áreas planas para a função
- Um ótimo local está situado na origem
- Ótimo global está no limite
- Se o procedimento de inicialização inicializa a população nos limites, este problema será resolvido facilmente.
- Uma depressão estreita para um ótimo global
- x ϵ [-5,5]^D, Ótimo global: $\boldsymbol{x}^* = \mathbf{o}_1$, $F_{20}(\boldsymbol{x}^*) = f\_bias_{20} = 10$

F21 - *Rotated Hybrid Composition Function*

$f_{1-2}(x)$: *Rotated Expanded Scaffer's F6 Function*

$$F(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2}) - 0.5}{1+0.001(x^2+y^2))^2}$$

$$f_i(x) = F(x_1, x_2) + F(x_2, x_3) + \ldots + F(x_{D-1}, x_D) + F(x_D, x_1)$$

$f_{3-4}(x)$: *Rastrigin's Function*

$$f_i(x) = \sum_{i=1}^{D}(x_1^2 - 10\cos(2\pi x_i) + 10)$$

$f_{5-6}(x)$: *F8F2 Function*

F8(x)= $\sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

F2(x)= $\sum_{i=1}^{D-1} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$

$f_i(x)$ = *F8(F2*(x_1, x_2)*)+ F8(F2*(x_2, x_3)*)+ …+ F8(F2* (x_{D-1}, x_D) + *F8(F2*(x_D, x_1)*))*

$f_{7-8}(x)$: *Weierstrass Function*

$f_i(x) = \sum_{i=1}^{D}(\sum_{k=0}^{kmax}[a^k \cos\left(2\pi b^k(x_i + 0.5)\right)]) - D\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k \cdot 0.5)]$,

a=0.5, b=3, $k_{max} = 20$

$f_{8-9}(x)$: *Griewank's Function*

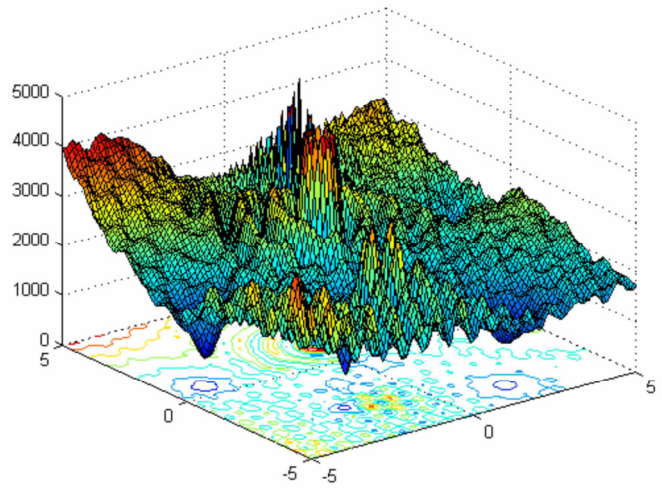
$f_i(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000}$ - $\prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

σ = [1,1,1,1,1,2,2,2,2,2] ,

λ = [5*5/100; 5/100; 5*1; 1; 5*1; 1; 5*10; 10; 5*5/200; 5/200];

$\boldsymbol{M}_i$ são todos matriz ortogonal

Figura 28 – Mapa 3-D para função F21 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

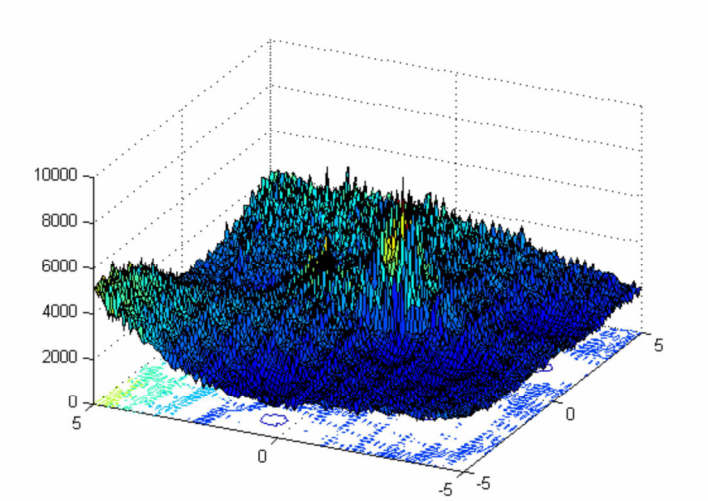
- Multimodal
- Rotacionada
- Não-separável
- Escalável
- Um grande número de ótimos locais

- Diferentes propriedades da função são misturadas
- x ϵ [-5,5]^D, Ótimo global: $\boldsymbol{x}^* = \mathbf{o}_1$, $F_{21}(\boldsymbol{x}^*) = f\_bias_{21} = 360$

F22 - *Rotated Hybrid Composition Function with High Condition Number Matrix*

Todas as configurações são as mesmas de F21exceto os números de condição M1 que são [10 20 50 100 200 1000 2000 3000 4000 5000]

Figura 29 – Mapa 3-D para função F22 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Ótimo global está no limite
- x ϵ [-5,5]^D, Ótimo global: $\boldsymbol{x}^* = \mathbf{o}_1$, $F_{22}(\boldsymbol{x}^*) = f\_bias_{22} = 360$

F23 - *Non-Continuous Rotated Hybrid Composition Function*

Todos os parametros são os mesmos de *F*21.

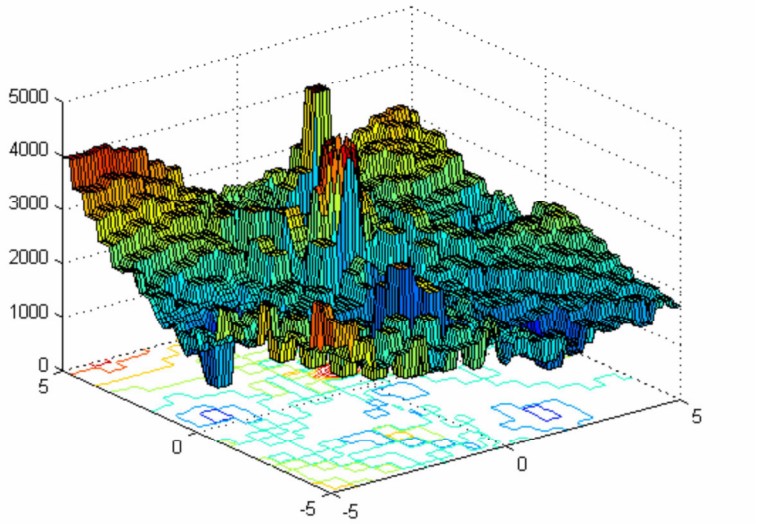
$$\text{Exceto } x_j = \begin{cases} x_j & |x_j - o_{1j}| < 1/2 \\ round(2x_j)/2 & |x_j - o_{1j}| \geq 1/2 \end{cases} \quad \text{para } j = 1,2,\dots,D$$

$$round(x) = \begin{cases} a-1 & if\ x \leq 0\ e\ b \geq 0{,}5 \\ a & if\ b < 0{,}5 \\ a+1 & if\ x > 0\ e\ b \geq 0{,}5 \end{cases}$$

Onde a é a parte inteira de x e b é a parte decimal de x

Todas as operações de rotação (round) neste texto usam esse mesmo esquema

Figura 30 – Mapa 3-D para função F23 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Não-contínua
- Ótimo global está no limite
- x ϵ [-5,5]^D, Ótimo global: $\boldsymbol{x}^* = \mathbf{o}_1$, $f(\boldsymbol{x}^*) = f\_bias_{(23)} = 360$

F24 - *Rotated Hybrid Composition Function* $f_1(x)$: *Weierstrass Function*

$f_i(x) = \sum_{i=1}^{D}(\sum_{k=0}^{kmax}[a^k \cos\left(2\pi b^k(x_i + 0.5)\right)]) - D\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k \cdot 0.5)]$,

a=0.5, b=3, $k_{max} = 20$

$f_2(x)$: *Rotated Expanded Scaffer's F6 Function*

$F(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2}) - 0.5}{1+0.001(x^2+y^2))^2}$

$f_i(x) = F(x_1, x_2) + F(x_2, x_3) + \ldots + F(x_{D-1}, x_D) + F(x_D, x_1)$

$f_3(x)$: *F8F2 Function*

$F8(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

$F2(x) = \sum_{i=1}^{D-1} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$

$f_i(x) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + \ldots + F8(F2(x_{D-1}, x_D) + F8(F2(x_D, x_1))$

$f_4(x)$: *Ackley's Function*

$f_i(x) = -20\ exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_1^2}) - exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$

$f_5(x)$: *Rastrigin's Function*

$f_i(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$

$f_6(x)$: *Griewank's Function*

$f_i(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

$f_7(x)$: *Non-Continuous Expanded Scaffer's F6 Function*

$F(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2}) - 0.5}{1+0.001(x^2+y^2))^2}$

$f(x) = F(y_1, y_2) + F(y_2, y_3) + \ldots + F(y_{D-1}, y_D) + F(y_D, y_1)$

$$y_j = \begin{cases} x_j & |x_j| < 1/2 \\ round(2x_j)/2 & |x_j| \geq 1/2 \end{cases} \quad \text{para } j = 1,2,\ldots,D$$

$f_8(x)$: *Non-Continuous Rastrigin's Function*

$f_i(x) = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 10)$

$$y_j = \begin{cases} x_j & |x_j| < 1/2 \\ round(2x_j)/2 & |x_j| \geq 1/2 \end{cases} \quad \text{para } j = 1,2,\ldots,D$$

$f_9(x)$: *High Conditioned Elliptic Function*

$f_i(x) = (\sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}}x_1^2$

$f_{10}(x)$: *Sphere Function with Noise in Fitness*

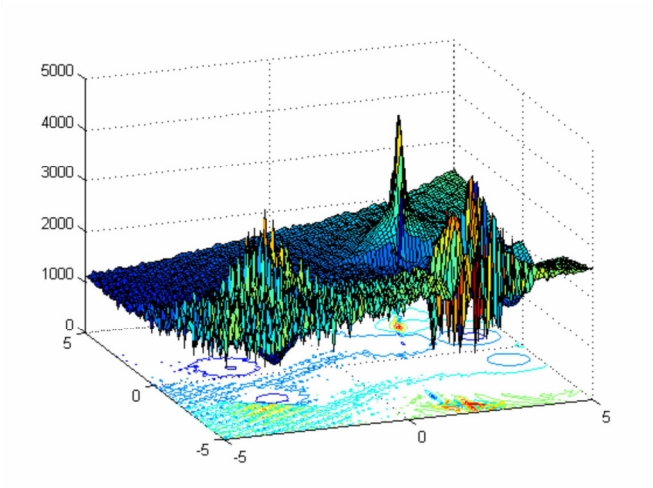
$f_i(x) = (\sum_{i=1}^{D} x_i^2)(1+0.1|N(0,1)|)$

σ_i = 2, para *i* = 1,2..., *D*

λ = [10; 5/20; 1; 5/32; 1; 5/100; 5/50; 1; 5/100; 5/100]

$\boldsymbol{M}_i$ são todas as matrizes de rotação. Números de condição são [100 50 30 10 5 5 4 3 2 2];

Figura 31 – Mapa 3-D para função F24 2-D



Fonte: Suganthan ET AL (2013)

Propriedades:

- Multimodal
- Rotacionada
- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Funções unimodais entregam áreas planas para a função
- x ϵ [-5,5]^D, Ótimo global: $\boldsymbol{x}^* = \mathbf{o}_1$, $F_{24}(\boldsymbol{x}^*) = f\_bias_{24} = 260$

F25 - *Rotated Hybrid Composition Function without Bounds*

Todas as configurações são as mesmas de F24, exceto por não ter interval de busca exato definido para essa função-teste

Propriedades:

- Multimodal

- Não-separável
- Escalável
- Um grande número de ótimos locais
- Diferentes propriedades da função são misturadas
- Funções unimodais entregam áreas planas para a função
- Ótimo global está no limite
- Não há limites

Inicializar população em $[2,5]^D$, Ótimo global: $\boldsymbol{x}^* = \mathbf{o}_1$ está fora do intervalo de inicialização, $F_{25}(\boldsymbol{x}^*) = f\_bias_{25} = 260$

4.2 RESULTADOS, GRÁFICOS E RANKING

No CEC 2005 foi aberta uma disputa, uma espécie de competição que iria avaliar diversos algoritmos, para tentar descobrir qual deles era a melhor meta-heurística de otimização existente. O modo de entrar na disputa era inscrever, através de um *paper*, uma meta-heurística que se propusesse a resolver as 25 funções-teste reunidas pelo comitê do evento. O *paper* deveria possuir pelo menos os nomes dos autores, um resumo, uma introdução à meta-heurística e ao próprio *paper*, a descrição da meta-heurística e do seu funcionamento, e os resultados dos testes de execução desses algoritmos nas suas tentativas de otimizar as funções dadas.

Inicialmente foi pedido que os algoritmos fossem executados com 10, 30 e com 50 dimensões, para cada um dos 25 problemas. Porém, às vésperas do evento, o mesmo comitê organizacional do CEC estipulou um limite de 8 páginas para cada *paper*, e para não extrapolar esse limite, os resultados dos testes para 50 dimensões poderiam ser omitidos. Por esse motivo, a maioria dos *papers* apresentados e aprovados pelo CEC não continham os testes para 50 dimensões, e por isso, apenas os resultados para 10 e para 30 dimensões foram considerados.

O teste realizado pelo CEC 2005 tinha as seguintes características:

- Otimização black-box das 25 funções dadas;
- 25 execuções para 10 dimensões e 25 para 30 dimensões, para cada algoritmo;

- Uma execução é considerada bem-sucedida se o ótimo global é alcançado com a precisão dada;
- O número máximo de execuções de função objetivo para 10 dimensões é 10^5, e para 30 dimensões é 3*10^5.

Nikolaus Hansen (2006), que inscreveu 2 algoritmos na competição (IPOP-CMA-ES e LR-CMA-ES), elaborou uma análise particular sobre os resultados dos testes, e entregou 6 gráficos demonstrando os seguintes resultados:

- Resultado geral (10 dimensões);
- Resultado geral (30 dimensões);
- Resultado com funções unimodais (10 dimensões);
- Resultado com funções unimodais (30 dimensões);
- Resultado com funções multimodais (10 dimensões);
- Resultado com funções multimodais (30 dimensões).

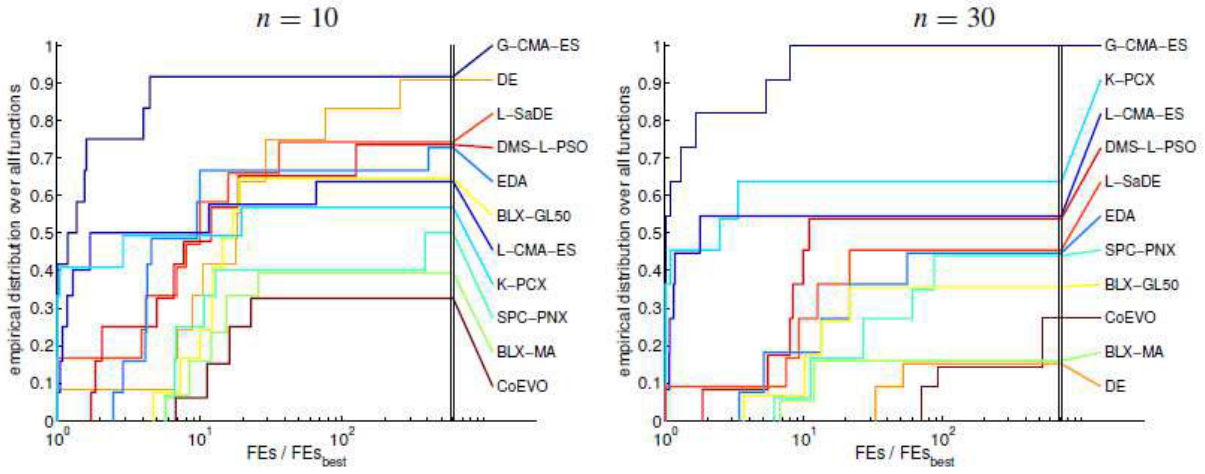
Além desses gráficos, Hansen também elaborou 6 quadros, que detalham ainda mais os resultados da sua análise. Os quadros contêm:

- Resultado com funções unimodais resolvidas (10 dimensões);
- Resultado com funções unimodais resolvidas (30 dimensões);
- Resultado com funções multimodais resolvidas (10 dimensões);
- Resultado com funções multimodais resolvidas (30 dimensões);
- Resultado com funções multimodais resolvidas (10 dimensões);
- Resultado com funções multimodais resolvidas (30 dimensões).

Ele considerou como funções resolvidas aquelas em que pelo menos um dos algoritmos conseguiu resolvê-la com sucesso em pelo menos uma execução, ou seja, dentro do limite de execuções permitido, e com resultado dentro da margem de erro aceitável, não importando se a função é unimodal ou multimodal, ou se é para 10 ou para 30 dimensões.

O gráfico do resultado geral obtido por Hansen, onde são comparados os desempenhos dos algoritmos ao resolver (otimizar) as funções, de acordo com os critérios de velocidade de convergência e quantidade de funções que o as algoritmos conseguiram solucionar, é apresentado na Figura 32, e cada algoritmo é representado por uma cor diferente. Deve-se atentar para o fato de a figura contem na verdade 2 gráficos, pois como foi explicado previamente, os testes foram separados para 10 e para 30 dimensões. A Figura 32 é a que segue:

Figura 32 – Gráfico do Resultado Geral



Fonte: Hansen (2006, não paginado)

Para se interpretar esses gráficos, devemos levar em conta que o eixo y (distribuição empírica sobre todas as funções) varia de 0 a 1, onde 1 representa 100% das funções resolvidas. Já o eixo x representa a razão entre a quantidade de avaliações feitas pelo algoritmo, e a quantidade de avaliações feitas pelo melhor algoritmo, ou seja, o eixo x representa a velocidade de convergência do algoritmo, ou melhor ainda, com que velocidade este algoritmo encontrou a solução. Altos valores no eixo y representa um bom resultado, enquanto que altos valores no eixo x representa uma alta lentidão na velocidade de conversão.

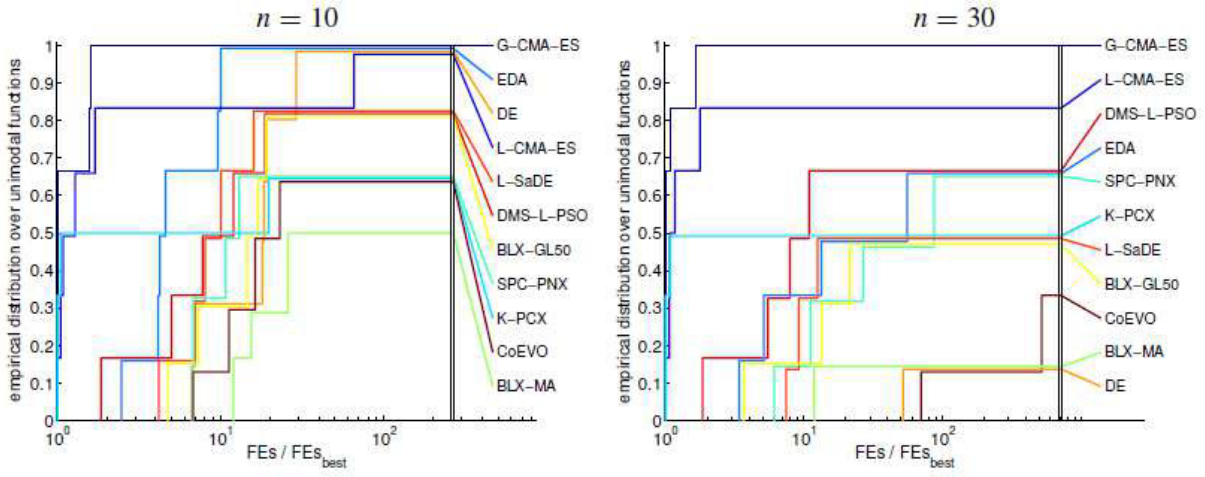
Portanto, pode-se inferir a partir desta figura 01 que, para n=10, onde n é o número de dimensões, os dois melhores algoritmos foram o G-CMA-ES e o DE, pois resolveram aproximadamente 90% das funções propostas, enquanto que os dois piores foram o BLX-MA com desempenho de aproximadamente 40% de capacidade de resolver os problemas dados, e o CoEVO, com pouco mais de 30%. Além disso, também pode-se inferir que o G-CMA-ES foi bem mais rápido do que o DE, por exemplo, porém não foi o mais rápido, e obteve aproximadamente 70% de lentidão em relação ao algoritmo que mais rápido convergiu para o ótimo global.

Para n=30, o G-CMA-ES continuou sendo o mais eficaz, e continuou não sendo o mais rápido, e em segundo lugar tem-se o L-CMA-ES, que resolveu um pouco mais de 80% das funções e levou um pouco mais de tempo para convergir que o G-CMA-ES. Já o DE, que foi o segundo melhor em solucionar problemas com

10 dimensões, teve uma queda absurda de desempenho, ficando próximo ao BLX-MA com apenas 15% de capacidade de solucionar os problemas.

Separando a análise por categoria de função, ou seja, unimodais resolvidas, multimodais resolvidas e multimodais não-resolvidas, temos o quadro 09 que exibe um Ranking dos algoritmos e a figura 33, que exibe os resultados obtidos por Hansen ao comparar os algoritmos durante o processo de otimização das funções unimodais apenas. Logo em seguida, tem-se os quadros 09 e 10, que exibem um resultado mais detalhado e um ranking dos algoritmos:

Figura 33 – Gráfico do Resultado para funções unimodais



Fonte: Hansen (2006, não paginado)

Como pode-se observar, ao solucionar apenas funções unimodais, o algoritmo G-CMA-ES continua liderando, tanto com 10 quanto com 30 dimensões. Já os piores desempenhos ficam por conta do algoritmo BLX-MA para 10 dimensões, e o DE para 30 dimensões. A seguir, é apresentado um quadro que demonstra um ranking elaborado por Hansen, onde é exibido a quantidade de funções solucionadas, o percentual de sucesso do algoritmo e um cálculo de desempenho individual de cada algoritmo para cada função, também duplicado porque apresenta resultados para 10 e 30 dimensões. Os valores inseridos na tabela são:

- Primeira linha: FEs = média normalizada de execuções da função, multiplicada pelo número total de execeuções (25) dividido pelo número de execuções bem sucedidas; sendo todos valores obtidos do algoritmo que apresentou o melhor resultado;

- Valores inseridos na tabela: FEs do algoritmo atual dividido pelo FEs do melhor algoritmo, sendo que entre parênteses é exibido os número de execuções bem sucedidas do algoritmo em questão, ou caso esse algoritmo não tenha conseguido resolver a função, é exibido entre colchetes a classificação do valor da média final da função.

Quadro 09 – Ranking dos algoritmos para funções Unimodais resolvidas

$n = 10$, $\mathrm{FE}_{\max} = 100000$

	solved functions	success rate	1 Sphere 1000	2 Schwefel 1.2 2400	3 Ellipsoid Condition 10^6 6500	4 Schwefel 1.2 with Noise 2900	5 Schwefel 2.6 on Bounds 5900	6 Rosenbrock 7100
G-CMA-ES	6	100%	1.6(25)	1.0(25)	1.0(25)	1.0(25)	1.0(25)	1.5(25)
EDA	6	97%	10.0(25)	4.6(25)	2.5(23)	4.1(25)	4.2(25)	9.6(22)
DE	6	96%	29.0(25)	19.2(25)	18.5(20)	17.9(25)	6.9(25)	6.6(24)
L-CMA-ES	6	88%	1.7(25)	1.1(25)	1.0(25)	65.5(7)	1.0(25)	1.3(25)
BLX-GL50	5	83%	19.0(25)	17.1(25)	[9]	14.5(25)	4.7(25)	7.3(25)
DMS-L-PSO	5	80%	12.0(25)	5.0(25)	1.8(25)	[11]	18.6(20)	7.7(25)
L-SaDE	5	77%	10.0(25)	4.2(25)	8.0(16)	15.9(24)	[9]	6.9(25)
SPC-PNX	4	67%	6.7(25)	12.9(25)	[11]	10.7(25)	6.8(25)	[10]
CoEVO	4	67%	23.0(25)	11.3(25)	6.8(25)	16.2(25)	[10]	[11]
K-PCX	4	62%	1.0(25)	1.0(25)	[8]	19.7(21)	[11]	1.0(22)
BLX-MA	3	49%	12.0(25)	15.4(25)	[10]	25.9(24)	[8]	[9]

$n = 30$, $\mathrm{FE}_{\max} = 300000$

	solved functions	success rate	1 Sphere 2700	2 Schwefel 1.2 12000	3 Ellipsoid Condition 10^6 43000	4 Schwefel 1.2 with Noise 59000	5 Schwefel 2.6 on Bounds 66000	6 Rosenbrock 60000
G-CMA-ES	6	90%	1.7(25)	1.1(25)	1.0(25)	1.0(10)	1.0(25)	1.0(25)
L-CMA-ES	5	83%	1.8(25)	1.2(25)	1.0(25)	[11]	1.1(25)	1.1(25)
EDA	4	67%	55.6(25)	13.3(25)	5.1(25)	3.4(25)	[3]	[10]
DMS-L-PSO	4	63%	1.9(25)	10.8(25)	7.9(21)	[9]	[9]	5.5(24)
BLX-GL50	3	50%	21.5(25)	13.3(25)	[7]	[6]	[6]	3.7(25)
SPC-PNX	4	45%	11.1(25)	26.7(22)	[11]	6.1(19)	[10]	86.7(1)
K-PCX	3	43%	1.0(25)	1.0(25)	[6]	[8]	[7]	1.1(14)
L-SaDE	3	41%	7.4(25)	12.5(24)	[8]	9.2(13)	[4]	[8]
BLX-MA	1	17%	11.9(25)	[10]	[10]	[7]	[7]	[8]
DE	1	17%	51.9(25)	[11]	[9]	[5]	[5]	[7]
CoEVO	2	7%	519 (3)	70.0(8)	[5]	[10]	[11]	[11]

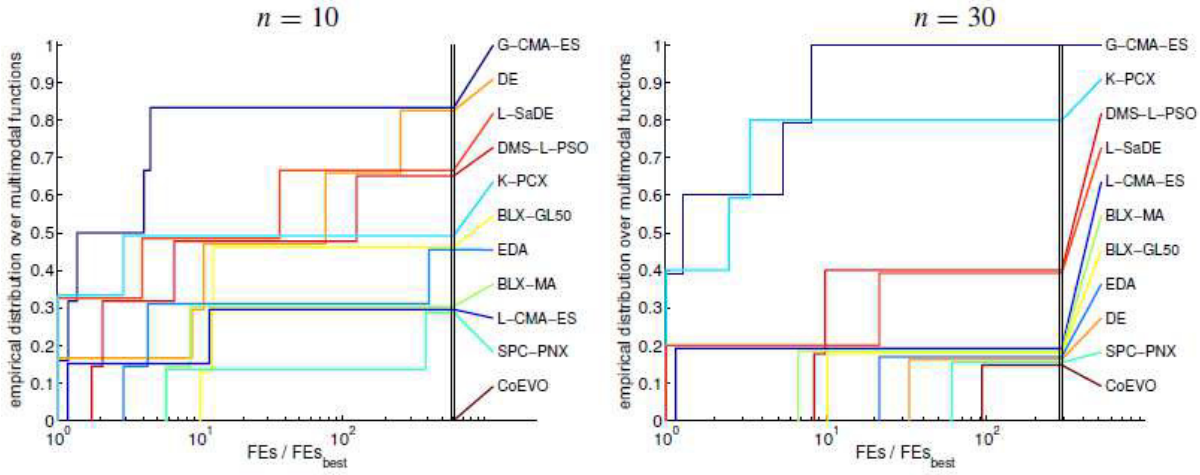
Fonte: Hansen (2006, não paginado)

Como se pode interpretar do quadro 09, para o caso de 10 dimensões, o algoritmo G-CMA-ES se saiu melhor, pois ele conseguiu otimizar 6 das 6 funções dadas, com uma taxa de 100% de sucesso, já que conseguiu solucionar a função

em 25 das 25 execuções, enquanto que o algoritmo que pior se saiu foi o BLX-MA, que só resolveu 3 funções das 6 dadas, com uma taxa de apenas 49% de sucesso, pois ele só conseguiu resolver 25 vezes as funções 1 e 2 para as 25 tentativas em cada uma das duas, e 24 vezes a função 4, nas 25 vezes que foi executado. As demais funções ele não conseguiu resolver dentro das condições definidas para os testes, e seu algoritmo foi o 10º, 8º e 9º para as funções 3, 5 e 6, respectivamente.

O mesmo ocorre para as funções multimodais, onde tem-se a figura 34, que exibe os gráficos dos desempenhos dos algoritmos, e o quadro

Figura 34 – Gráfico do Resultado para funções multimodais



Fonte: Hansen (2006, não paginado)

Quadro 10 – Ranking dos algoritmos para funções multimodais resolvidas

$n = 10$, $\mathrm{FE}_{\max} = 100000$

	solved functions	success rate	7 Griewank out Bounds 4700	9 Rastrigin Separable 17000	10 Rastrigin Rotated 55000	11 Weierstrass 190000	12 Schwefel 2.13 8200	15 Hybrid Separable 33000
G-CMA-ES	5	63%	1.0(25)	4.5(19)	1.2(23)	1.4(6)	4.0(22)	[6]
L-SaDE	4	53%	36.2(6)	1.0(25)	[5]	[8]	3.9(25)	1.0(23)
DMS-L-PSO	4	47%	126 (4)	2.1(25)	[3]	[7]	6.6(19)	1.7(22)
K-PCX	3	40%	[10]	2.9(24)	1.0(22)	[10]	1.0(14)	[11]
DE	5	30%	255 (2)	10.6(11)	[9]	1.0(12)	8.8(19)	75.8(1)
L-CMA-ES	2	25%	1.2(25)	[11]	[10]	[6]	11.6(12)	[6]
BLX-GL50	3	17%	12.3(9)	10.0(3)	[5]	[5]	12.1(13)	[9]
BLX-MA	2	15%	[11]	5.7(18)	[7]	[9]	[9]	8.5(5)
EDA	3	9%	404 (1)	[9]	[4]	2.9(3)	4.3(10)	[9]
SPC-PNX	2	1%	383 (1)	[8]	[8]	5.8(1)	[10]	[6]
CoEVO	0	0%	[9]	[10]	[11]	[11]	[11]	[8]

$n = 30$, $FE_{max} = 300000$

	solved functions	success rate	6100	99000	450000	5000000	180000	
			7 Griewank out Bounds	9 Rastrigin Separable	10 Rastrigin Rotated	11 Weierstrass	12 Schwefel 2.13	15 Hybrid Separable
K-PCX	4	38%	2.5(10)	3.3(18)	1.0(14)	[7]	1.0(5)	[11]
G-CMA-ES	5	37%	1.0(25)	8.0(9)	5.3(3)	1.0(1)	1.3(8)	[1]
L-SaDE	2	36%	21.3(20)	1.0(25)	[4]	[5]	[4]	[2]
DMS-L-PSO	2	22%	9.8(24)	[6]	[5]	[5]	8.3(4)	[4]
EDA	1	20%	21.3(25)	[10]	[9]	[10]	[7]	[4]
BLX-GL50	1	20%	10.2(25)	[5]	[3]	[4]	[8]	[4]
DE	1	20%	32.8(22)	[6]	[6]	[10]	[5]	[8]
L-CMA-ES	1	20%	1.1(25)	[11]	[11]	[3]	[9]	[8]
SPC-PNX	1	13%	60.7(16)	[8]	[7]	[2]	[10]	[8]
CoEVO	1	9%	93.4(11)	[9]	[10]	[9]	[11]	[10]
BLX-MA	1	7%	[11]	6.7(9)	[8]	[8]	[6]	[4]

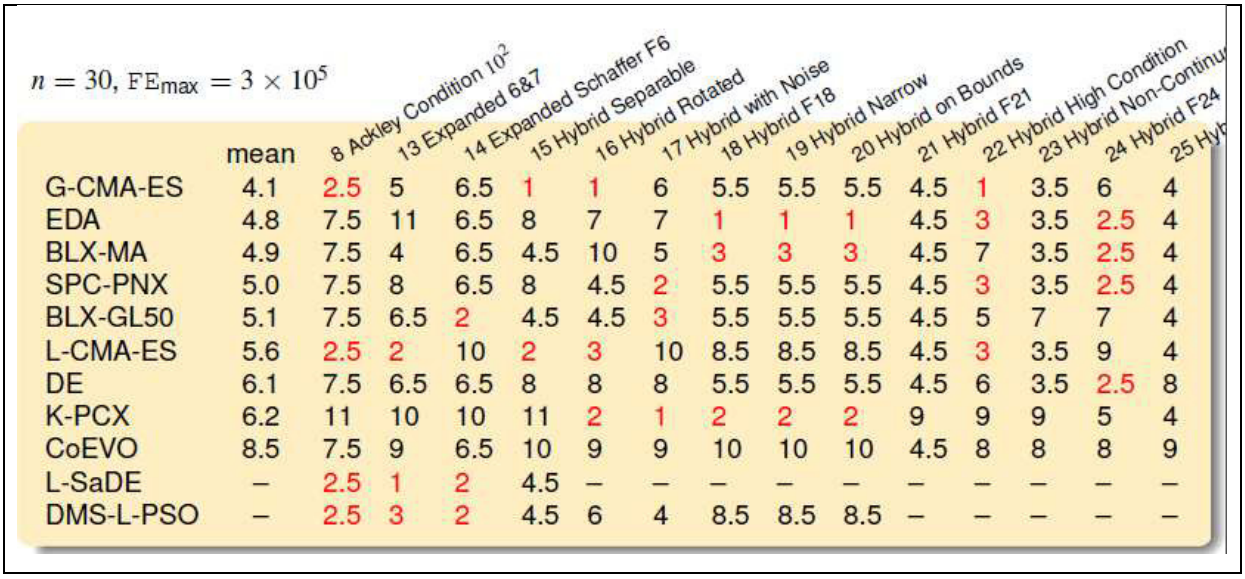
Fonte: Hansen (2006, não paginado)

Além dessas, ainda houve as funções multimodais que nenhum dos algoritmos conseguiu resolver dentro dos critérios estabelecidos para a competição no CEC 2005. A seguir tem-se o quadro 11 que exibe os resultados nestes casos, lembrando que nessa tabela, os algoritmos foram classificados de acordo com suas médias finais para cada função:

Quadro 11 – Ranking dos algoritmos para funções multimodais não resolvidas

$n = 10$, $FE_{max} = 10^5$

	mean	8 Ackley Condition 10^2	13 Expanded 6&7	14 Expanded Schaffer F6	16 Hybrid Rotated	17 Hybrid with Noise	18 Hybrid F18	19 Hybrid Narrow	20 Hybrid on Bounds	21 Hybrid F21	22 Hybrid High Condition	23 Hybrid Non-Continuous	24 Hybrid F24	25 Hybrid out Bo
G-CMA-ES	3.8	5.5	4	7	1	5.5	3	2.5	2.5	5.5	1.5	4	5	2.5
BLX-GL50	4.2	5.5	6	1.5	2.5	3	3	2.5	2.5	5.5	5.5	4	5	8.5
L-SaDE	5.3	5.5	1	6	5.5	3	8	7.5	7.5	5.5	5.5	4	5	4.5
DMS-L-PSO	5.7	5.5	2	3.5	4	3	8	7.5	7.5	5.5	5.5	8	5	8.5
L-CMA-ES	6.0	5.5	3	11	7.5	11	6	5	5	1	3	4	11	4.5
SPC-PNX	6.2	11	8	8	7.5	5.5	3	2.5	2.5	5.5	9	4	5	8.5
EDA	6.2	5.5	10	5	9	8	3	7.5	7.5	5.5	8	4	5	2.5
BLX-MA	6.6	5.5	7	1.5	5.5	7	8	7.5	7.5	10	5.5	10	5	6
K-PCX	7.0	5.5	5	3.5	2.5	1	10	11	10	11	1.5	11	10	8.5
DE	7.0	5.5	11	10	11	10	3	2.5	2.5	5.5	10	4	5	11
CoEVO	8.2	5.5	9	9	10	9	11	10	11	5.5	11	9	5	1



$n = 30$, $FE_{max} = 3 \times 10^5$

	mean	8 Ackley Condition 10^2	13 Expanded 6&7	14 Expanded Schaffer F6	15 Hybrid Separable	16 Hybrid Rotated	17 Hybrid with Noise	18 Hybrid F18	19 Hybrid Narrow	20 Hybrid on Bounds	21 Hybrid F21	22 Hybrid High Condition	23 Hybrid Non-Continu	24 Hybrid F24	25 Hyb
G-CMA-ES	4.1	2.5	5	6.5	1	1	6	5.5	5.5	5.5	4.5	1	3.5	6	4
EDA	4.8	7.5	11	6.5	8	7	7	1	1	1	4.5	3	3.5	2.5	4
BLX-MA	4.9	7.5	4	6.5	4.5	10	5	3	3	3	4.5	7	3.5	2.5	4
SPC-PNX	5.0	7.5	8	6.5	8	4.5	2	5.5	5.5	5.5	4.5	3	3.5	2.5	4
BLX-GL50	5.1	7.5	6.5	2	4.5	4.5	3	5.5	5.5	5.5	4.5	5	7	7	4
L-CMA-ES	5.6	2.5	2	10	2	3	10	8.5	8.5	8.5	4.5	3	3.5	9	4
DE	6.1	7.5	6.5	6.5	8	8	8	5.5	5.5	5.5	4.5	6	3.5	2.5	8
K-PCX	6.2	11	10	10	11	2	1	2	2	2	9	9	9	5	4
CoEVO	8.5	7.5	9	6.5	10	9	9	10	10	10	4.5	8	8	8	9
L-SaDE	–	2.5	1	2	4.5	–	–	–	–	–	–	–	–	–	–
DMS-L-PSO	–	2.5	3	2	4.5	6	4	8.5	8.5	8.5	–	–	–	–	–

Fonte: Hansen (2006, não paginado)

Para chegar a uma conclusão geral sobre todos os resultados e em todas as categorias, o autor desta monografia elaborou um esquema de atribuição de pontos, inversamente proporcional à sua classificação nos quadros, ou seja, o primeiro colocado recebe 11 pontos, enquanto que o último recebe apenas 1 ponto. Ao final, essa pontuação é somada e os algoritmos são classificados de 1 a 11.

Quadro 12 – Ranking Geral das meta-heurísticas de otimização do CEC 2005

Algoritmo	UMR 10D	UMR 30D	MMR 10D	MMR 30D	MMNR 10D	MMNR 30D	Total de pontos
BLX-GL50	7	7	5	6	10	7	42
BLX-MA	1	3	4	1	4	9	22
CoEVO	3	1	1	2	1	3	11
DE	9	2	7	5	2	5	30
DMS-L-PSO	6	8	9	8	8	1	40
EDA	10	9	3	7	5	10	44
G-CMA-ES	11	11	11	10	11	11	65
K-PCX	2	5	8	11	3	4	33
L-CMA-ES	8	10	6	4	7	6	41
L-SaDE	5	4	10	9	9	2	39
SPC-PNX	4	6	2	3	6	8	29

Fonte: Elaborado pelo autor (2013)

Com base neste cálculo, a ordem de classificação dos algoritmos por pontuação é:

Quadro 13 – Classificação Final

Classificação	Algoritmo	Pontuação
1º	G-CMA-ES	65
2º	EDA	44
3º	BLX-GL50	42
4º	L-CMA-ES	41
5º	DMS-L-PSO	40
6º	L-SaDE	39
7º	K-PCX	33
8º	DE	30
9º	SPC-PNX	29
10º	BLX-MA	22
11º	CoEVO	11

Fonte: Elaborado pelo autor (2013)

A partir de todos os dados coletados neste trabalho, pode-se inferir que a meta-heurística campeã, de maneira expressiva, atingindo 65 pontos de 66 possíveis, perdendo apenas para o algoritmo K-PCX nas funções multimodais resolvidas, para 30 dimensões.

4.3 CONSIDERAÇÕES FINAIS

O campeonato mundial de meta-heurísticas evolutivas foi um desafio muito interessante tanto para os organizadores do evento quanto para os desenvolvedores dos algoritmos de otimização competidores, pois os organizadores tiveram a difícil missão de elaborar um conjunto de testes e métricas que pudessem medir de forma justa e eficiente algoritmos tão diferentes, ao mesmo tempo que os desenvolvedores dos algoritmos que competiram tiveram que elaborar algoritmos que atendessem a problemas totalmente diferentes um do outro, sem que fosse

preciso ajustar o algoritmo a cada um dos problemas, realizando no máximo, um pequeno ajuste de parâmetros.

A pesquisa feita acerca do tema possuía pelo menos duas hipóteses a serem verificadas: se era possível avaliar o desempenho de diversos algoritmos entre si e, caso fosse possível, qual seria o melhor algoritmo de otimização contínua existente. A pesquisa realizada neste trabalho mostrou que sim, é possível comparar desempenhos de algoritmos de otimização, e consequentemente, um algoritmo pôde ser apontado como o melhor, que foi o G-CMA-ES (ou IPOP-CMA-ES).

A pesquisa foi totalmente baseada em resultados de estudos de autores e alguns materiais de apoio para fornecer mais conteúdo e uma maior clareza ao conteúdo exposto. Como foi um tema bem específico, porém de um conteúdo muito amplo, não foi uma tarefa simples encontrar materiais disponíveis, o que inviabilizou inclusive, de ser feito um trabalho com conteúdo de data mais recente, por exemplo, o trabalho poderia apresentar conteúdo resultantes do CEC 2013, que foi o mais recente, porém até mesmo os materiais disponíveis sobre o próprio CEC são difíceis de encontrar. Um dos motivos que tornam esse conteúdo algo de difícil acesso é justamente o fato de o CEC ser realizado cada ano em um local diferente, e organizado por uma instituição diferente em cada ano, porque embora o CEC seja um evento do IEEE, quem fornece estrutura ao evento é uma instituição educacional diferente a cada ano. Sendo assim, os materiais sobre o CEC praticamente se perdem, e fica quase impossível de encontra-los.

Outra dificuldade também inerente a pesquisar sobre este tema relacionando-o ao CEC, é que o CEC não é um congresso que aborda somente esse tema, de otimização. O CEC também aborda outros temas variados à computação, abrindo uma gama de resultados de busca que não interessam a que busca especificamente conteúdo sobre Otimização Contínua e Meta-heurísticas evolutivas.

Ainda assim, com todas essas dificuldades encontradas, foi possível se fazer uma pesquisa muito satisfatória a respeito do tema, e que gerou resultados também satisfatórios. Foi mostrado que é possível comparar meta-heurísticas evolutivas de otimização de funções com domínio contínuo, bastando para isso, elaborar um conjunto de métricas adequado para tal medição e, em consequência disto, foi possível elaborar um *ranking*, mesmo que de uma forma razoavelmente simplória, sobre os algoritmos que foram inscritos na competição, para serem

avaliados e comparados entre si. Em compensação, foi possível obter informações que primeiramente, não faziam parte dos objetivos principais da pesquisa, porém se tornaram algo que não deve ser oculto.

Não é porque o algoritmo G-CMA-ES venceu esta competição que ele pode ser dito como a melhor meta-heurística absoluta. Ele foi superado em velocidade de conversão por diversas vezes durante a competição. Caso o número limite de avaliações fosse reduzido, justamente por ele ser mais lento, ele poderia não ser capaz de encontrar a resposta a tempo e isto acarretaria numa considerável mudança de sua posição no ranking. Também existem casos de algoritmos que seria beneficiados com o aumento do tempo de avaliação, como é o caso do DE, que no teste de funções unimodais com 10 dimensões, foi o 3º colocado, porém nas mesmas funções, aumentando as dimensões para 30, o mesmo algoritmo despencou para a penúltima posição. Este fato não quer dizer que ele seja ineficaz, apenas pode ser resultado de que ele precisaria de mais tempo ou de melhor ajuste para os problemas que ele iria solucionar.

Outra fato notável diz respeito ao CoEVO, que é um algoritmo com um raciocínio muito interessante, mas que infelizmente não obteve bons resultados. O motivo que parece ser a falha dele é que na etapa de seleção de solução a ser substituída, ele faz essa seleção de forma randômica, o que teoricamente era para lhe prevenir a convergência prematura, no entanto, ele sofreu com muita lentidão no processo de convergência e foi facilmente enganado por ótimos locais, o que lhe tirava do objetivo de encontrar o ótimo global. Talvez uma versão melhorada do CoEVo, com esse processo de seleção sendo melhorado, e sendo inserido um processo mais eficaz de espalhar a população pelo espaço de busca, pudesse leva-lo a uma posição melhor no *ranking*.

Também é interessante quando se compara algoritmos bem parecidos que diferem em um ou outro pequeno detalhe, como é o caso do BLX-GL50 e o BLX-MA. Os dois são quase o mesmo algoritmo, rodam um algoritmo com característica global para explorar a região de busca, e em cima desse resultado, rodam outro algoritmo de busca local para intensificar a busca. O algoritmo de característica global é o mesmo algoritmo genético para os dois, porém o algoritmo de característica local escolhido por cada um dos dois foi quem fez a diferença de 7 posições entre eles no ranking, deixando o GL50 em 3º lugar e o MA em 10º. Resultado semelhante ocorreu entreos dois algoritmos baseados em CMA-ES que

disputaram. O G-CMA-ES é praticamente o mesmo algoritmo do L-CMA-ES, porém o G-CMA-ES aumenta a população a cada iteração do algoritmo, conferindo a ele uma forte característica global, enquanto que o L-CMA-ES não o faz e ainda utiliza um algoritmo de busca local para melhorar seu comportamento como otimizador local. Este fator foi decisivo para fazer do G-CMA-ES o campeão da competição enquanto que o L-CMA-ES ficou com apenas um modesto 4º lugar.

Outro caso curioso foi entre o DE e o L-SaDE, que é uma versão de DE toda modificada, feita para melhorar o desempenho do DE, eliminando pontos considerados como falhas no algoritmo do seu antecessor. Apesar de ter a intenção de superar o DE corrigindo suas falhas, e de tê-lo conseguido em alguns casos, inacreditavelmente o DE o superou em outras situações, talvez justamente por ter um “raciocínio” mais simples e consequentemente mais rápido.

Por fim, pode-se apontar o G-CMA-ES como o vencedor da competição por ter sua eficácia comprovada nos diversos testes, e pelo fato de o mesmo ter se mostrado superior aos outros concorrentes em quase todos os testes, porém ele pode ter uma implementação complicada devido aos extensos cálculos envolvidos em seu interior. Por isso não se pode afirmar que ele é definitivamente melhor que o DE, por exemplo, já que este é muito mais fácil para ser implementado e pode atender perfeitamente às necessidades do usuário.

5 CONCLUSÃO

Baseando-se nos resultados obtidos do CEC 2005, pôde-se concluir que foi possível estabelecer um comparativo entre as meta-heurísticas existentes, e sendo assim, pôde-se também apontar qual foi a melhor meta-heurística existente na atualidade, que é o IPOP-CMA-ES (ou G-CMA-ES).

Pôde-se chegar também à conclusão de que mesmo se tendo apontado um algoritmo como sendo a melhor meta-heurística existente na atualidade, isso não quer dizer que esse algoritmo será efetivamente o melhor sempre, pois outros fatores devem ser levados em conta quando se transfere essa teoria para a prática: custo de implementação, disponibilidade de recursos computacionais, velocidade, exatidão nos resultados, etc. O trabalho aponta que o IPOP-CMA-ES é muito complicado para se implementar, devido aos seus extensos cálculos relacionados à sua matriz de covariância. Além disso, ele também não foi o mais rápido em todos os casos. Então, para uma função unimodal simples qualquer, um algoritmo DE pode ser mais útil do que o IPOP-CMA-ES, por ser muito mais simples de implementar, eficiente em várias áreas de aplicação, rápido e com uma boa precisão.

Conclui-se portanto que, é extremamente importante analisar bem a situação antes de determinar qual meta-heurística será aplicada no propósito desejado.

Como trabalhos futuros, fica a possibilidade de implementar algumas dessas meta-heurísticas, e testá-las não com essas funções reunidas pelo CEC 2005, mas com funções utilizadas no dia a dia, com aplicações reais, em variados âmbitos e áreas de aplicação, como nas engenharias, ciências, linhas de produção, sistemas de tráfego, etc.

Este trabalho visa também, orientar profissionais com interesse na área, servindo como um guia que os permita escolher de forma mais eficaz qual meta-heurística utilizar, ou em quais tipos de problema aplicar cada uma delas. Também seria interessante, caso houvesse mais tempo e recursos disponíveis, implementar estas meta-heurísticas e testá-las alterando seus parâmetros e os parâmetros de teste, para verificar se algo mudaria no resultado, como se é levado a crer.

REFERENCIAS

AUGER, Anne; HANSEN, Nikolaus. **Performance Evaluation of an Advanced Local Search Evolutionary Algorithm.** Evolutionary Computation, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005. p. 1777-1784. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/auger05pea.pdf> Acesso em: 05 ago. 2013

AUGER, Anne; HANSEN, Nikolaus. **A Restart CMA Evolution Strategy With Increasing Population Size.** p. 1769-1776. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/auger05ARCMA.pdf> Acesso em: 05 ago. 2013

BECCENERI, José Carlos. **Meta-heurísticas bio-inspiradas aplicadas a problemas de otimização.** Disponível em: <http://www.lac.inpe.br/~becceneri/palestras/Meta-heuristicas.pdf> Acesso em: 31 jul. 2013.

BENNATON, Jocelyn Freitas. **O que é otimização?** Abr. 2001. Disponível em: <http://www.ebah.com.br/content/ABAAAfWZ8AH/que-otimizar > Acesso em: 31 jul. 2013.

BOCCATO, Levy; ATTUX, R. R. de F.; ZUBEN, F. J. V. **Evolução Diferencial:** Introdução e Conceitos Básicos. Jun. 2009. DCA – UNICAMP. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_1s11/notas_de_aula/topico12_IA707_1s11.pdf> Acesso em: 22 set. 2013

BUENO, Fabricio. **Métodos Heurísticos Teorias e Implementações.** Disponível em: <https://wiki.ifsc.edu.br/mediawiki/images/b/b7/Tutorial_m%C3%A9todos_heur%C3%ADsticos.pdf> Acesso em: 31 jul. 2013.

FINGLER, Henrique. **Otimização de colônias de formigas em cuda:** o problema da mochila e o problema quadrático de alocação. Abr. 2013. 73 f. FACOM – Faculdade de computação. UFMS – Universidade Federal do Mato Grosso de Sul. Disponível em: <http://bdtd.cbc.ufms.br/tedesimplificado/tde_arquivos/1/TDE-2013-04-30T085633Z-972/Publico/Henrique%20Fingler.pdf> Acesso em: 01 out. 2013.

GALLAGHER, K. ET AL. **Real-Parameter Optimization Performance Study on the CEC-2005 benchmark with SPC-PNX.** Evolutionary Computation, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005. p. 498-505. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/ballester05rpo.pdf> Acesso em: 05 ago. 2013.

GARCÍA-MARTINEZ, C. LOZANO, M. **Hybrid Real-Coded Genetic Algorithms with Female and Male Differentiation.** Evolutionary Computation, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005. p. 896-903. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/garcia-martinez05hrc.pdf> Acesso em: 05 ago. 2013.

HANSEN, Nikolaus. **Compilation of Results on the 2005 CEC Benchmark Function Set**. Computational Laboratory (CoLab) - Institute of Computational Science. ETH Zurich. 04 May 2006. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/hansen05compareresults.pdf> Acesso em: 15 ago. 2013.

LIANG, J. J.; SUGANTHAN, P. N. **Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search.**Evolutionary Computation, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005. p. 522-528. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/liang05dms.pdf> Acesso em: 05 ago. 2013.

METAHEURISTICS NETWORK. Disponível em: <http://www.metaheuristics.net/> Acesso em: 22 ago. 2013.

MIYAZAWA, Flavio Keidi. Campinas, ? 2008. **Heurísticas e Metaheurísticas**. Aulas proferidas no Instituto de Computação da Universidade de Campinas – UNICAMP. Disponível em: <http://www.ic.unicamp.br/~fkm/disciplinas/mc658/2013s1/slides/aula08.pdf> Acesso em: 22 ago. 2013.

MOLINA, Daniel; HERRERA, Francisco; LOZANO, Manuel. **Adaptive Local Search Parameters for Real-Coded Memetic Algorithms.** Evolutionary Computation, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005 Page(s): 888-895. Disponível em:< http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/molina05als.pdf> Acesso em: 05 ago. 2013.

POŠÍK, Petr. **Real Parameter Optimisation Using Mutation Step Co-evolution**. Evolutionary Computation, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005. p. 872-879. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/posik05rpo.pdf> Acesso em: 05 ago. 2013.

PRICE, Kenneth; STORN, Rainer. **Differential Evolution (DE):** for Continuous Function Optimization. Disponível em: <http://www1.icsi.berkeley.edu/~storn/code.html> Acesso em: 07 ago. 2013.

QUIN, A.K.; SUGANTHAN, P. N. **Self-adaptive Differential Evolution Algorithm for Numerical Optimization.** Evolutionary Computation, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005. p. 1785-1791. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/quin05sad.pdf> Acesso em: 05. ago. 2013.

RÖNKKÖNEN, Jani; KUKKONEN, Saku; PRICE, Kenneth V. Real-Parameter **Optimization with Differential Evolution. Evolutionary Computation**, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005. p. 506-513. Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/ronkkonen05rpo.pdf> Acesso em: 05 ago. 2013.

SINHA, Ankur; TIWARI, Santosh; KALYANMOY, Deb. **A Population-Based, Steady-State Procedure for Real-Parameter Optimization.** p. 514-521. Disponível

em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/sinha05apb.pd> Acesso em: 05. Ago. 2013.

SUGANTHAN, P.N.; ET AL. **Problem Definitions and Evaluation Criteria for the CEC 2005**: Special Session on Real-Parameter Optimization. May, 2005. Technical Report, Nanyang Technological University, Singapore and KanGAL Report Number 2005005 (Kanpur Genetic Algorithms Laboratory, IIT Kanpur). Disponível em: <https://www.lri.fr/~hansen/Tech-Report-May-30-05.pdf> Acesso em: 07 set. 2013.

TALBI, El-Ghazali. **Metaheuristics:** From design to implementation.
John Wiley & Sons, 2009.

WIKIPEDIA. **Otimização**. Disponível em: <http://pt.wikipedia.org/wiki/Otimiza%C3%A7%C3%A3o > Acesso em: 31 jul. 2013a.

WIKIPEDIA. **Estimation of distribution algorithm**. Disponível em: <http://en.wikipedia.org/wiki/Estimation_of_distribution_algorithm> Acesso em: 10 ago. 2013b.

WIKIPEDIA.**CMA-ES.** Disponível em: <http://en.wikipedia.org/wiki/CMA-ES> Acesso em: 02 set. 2013c.

YUAN, Bo; GALLAGHER, Marcus. **Experimental Results for the Special Session on Real-Parameter Optimization at CEC 2005:** A Simple, Continuous EDA. Evolutionary Computation, 2005. The 2005 IEEE Congress on 2-5 Sept. 2005. p. 1792-1799.Disponível em: <http://sci2s.ugr.es/eamhco/pdfs/contributionsCEC05/yuan05erf.pdf> Acesso em: 05 ago. 2013.