Alfredo José de Paula Barbosa

Algoritmo de Busca Gravitacional com a Constante Gravitacional Normalizada

Itajubá, Minas Gerais
23 de julho de 2018

Alfredo José de Paula Barbosa

Algoritmo de Busca Gravitacional com a Constante Gravitacional Normalizada

Universidade Federal de Itajubá
Instituto de Engenharia de Sistemas e Tecnologia da Informação
Ciência e Tecnologia da Informação (Mestrado)

Orientador: Otávio Augusto Salgado Carpinteiro
Coorientador: Carlos Henrique Valério de Moraes

Itajubá, Minas Gerais
23 de julho de 2018

Eu dedico este trabalho à minha mulher, Geyziane Monteiro Barbosa, que tem me acompanhado e suportado.

Agradecimentos

Eu agradeço aos meus pais, que me deram a vida; aos meus professores, que me prepararam para essa pesquisa; e à CAPES, que financiou essa pesquisa. Sobretudo, que Deus seja louvado.

“Bem sei eu que tudo podes, e que nenhum dos teus propósitos pode ser impedido.”
(Jó 42:2, Bíblia Sagrada)

Resumo

O Algoritmo de Busca Gravitacional é um algoritmo de otimização global baseado nas leis da gravidade e da dinâmica de Newton. O algoritmo já está bem difundido, tendo qualidade de convergência melhor que o algoritmo genético e enxame de partículas em inúmeras aplicações, com várias versões presentes nas literaturas recentes. O método possui generalidade e confiabilidade já demonstradas, mas sua qualidade de convergência é sensivelmente afetada pela má escolha do valor inicial de sua constante gravitacional. O grande problema é que até o momento não existia uma fórmula nem método para definir um valor apropriado para a constante gravitacional inicial, comprometendo a eficácia e tempo de convergência do algoritmo. Assim, é proposta uma heurística baseada na teoria de Brans-Dicke, para determinar a constante gravitacional inicial conforme o espaço de busca do problema. Confirmou-se a eficácia da proposta realizando uma série de otimizações em funções de referência, com resultados superiores ao algoritmo original, destacando a qualidade da solução para espaços extremamente irregulares. Esta nova proposta é chamada de *Gravitational Search Algorithm with Normalized Gravitational Constant*.

Palavras-chaves: algoritmos de otimização. otimização global. algoritmo de busca gravitacional.

Abstract

Gravitational Search Algorithm is a global optimization algorithm based on Newton's gravity and dynamics laws. The algorithm is well known, having a better convergence than genetic and particle swarm optimization algorithms, in various applications in the recent literature. The method's generality and dependability has already been demonstrated, but its convergence quality is sensibly affected by poor choices of the gravitational constant's initial value. The major problem is there wasn't a formula or method to define an apropriate value for the initial gravitational constant so far, compromising the efficacy and convergence speed of the algorithm. So it's proposed a heuristic based on the Brans-Dicke Theory, to determine the initial gravitational constant according to the problem's search space. The efficacy of this heuristic is confirmed by multiple benchmark functions optimizations, with results superior to the original algorithm, highlighting the solution quality for extremely irregular spaces. This new proposal is named Gravitational Search Algorithm with Normalized Gravitational Constant.

Keywords: *optimization algorithms. global optimization. gravitational search algorithm.*

Lista de abreviaturas e siglas

ANN	*Artificial Neural Network*
BBO	*Biogeography-Based Optimization*
BGSA	*Binary Gravitational Search Algorithm*
BH-GSA	*Black Hole-Gravitational Search Algorithm*
BPSO	*Binary Particle Swarm Optimization*
CGGSA	*Cauchy and Gaussian Mutation Gravitational Search Algorithm*
CGSA	*Cumulative Gravitational Search Algorithm*
DGSA	*Discrete Gravitational Search Algorithm*
FCGSA	*Fully Constrained Gravitational Search Algorithm*
FGSA	*Fuzzy Gravitational Search Algorithm*
FVGGSA	*Fitness Varying Gravitational Constant Gravitational Search Algorithm*
FOA	*Fuzzy Optimal Assessment*
GA	*Genetic Algorithm*
GSA	*Gravitational Search Algorithm*
GSA-NGC	*Gravitational Search Algorithm with Normalized Gravitational Constant*
GSAN	*Gravitational Search Algorithm with Negative Mass*
HC	*Hill Climbing*
IGSA	*Improved Gravitational Search Algorithm*
MGSA	*Multi-Period Gravitational Search Algorithm*
MGSA-NSGA-III	*Modified Gravitational Search Algorithm based on Non-Dominated Sorting Genetic Algorithm III*
MO-GSA	*Multi-Objective Gravitational Search Algorithm*
NGC	*Normalized Gravitational Constant*

NGSA *Niche Gravitational Search Algorithm*

PCGSA *Partially Constrained Gravitational Search Algorithm*

PSO *Particle Swarm Optimization*

PSOGSA *Particle Swarm Optimization Gravitational Search Algorithm*

RS *Random Search*

SA *Simulated Annealing*

SGSA *Standard Gravitational Search Algorithm*

SHC *Stochastic Hill Climbing*

TVAC-GSA-PSO *Time Varying Acceleration Coefficients-Gravitational Search Algorithm-Particle Swarm Optimization*

UGSA *Unconstrained Gravitational Search Algorithm*

Lista de ilustrações

Lista de Algoritmos

Lista de Códigos

Lista de tabelas

Sumário

1 Introdução

1.1 Problemas de Otimização

O problema de otimização é uma modelagem matemática de um problema de decisão. O objetivo de um problema de otimização é a maximização ou minimização de uma função objetiva (*objective function*). A função objetiva é uma função $f : \mathbb{R}^N \rightarrow \mathbb{R}$, onde $y = f(x)$ é a medida da qualidade da solução x, composta de N variáveis. Ou seja, a função objetiva é proporcional à qualidade da solução. Considerando um problema de maximização, x^* é uma solução ótima de $f(x)$ se, e somente se, se $f(x^*) \geq f(x)$ para todo x, e vice-versa. Além disso, se x^* é uma solução ótima de $f(x)$, então $f^* = f(x^*)$ é o valor ótimo de $f(x)$.

Algumas funções objetivas são sujeitas a restrições, ou seja, algumas soluções não podem ser consideradas na prática. As restrições funcionais (*functional constraints*) da função objetiva são indicadas pelas equações: $g_i(x) = 0,\ i = 1,\ 2,\ ...,\ M$ e inequações: $h_j(x) \geq 0,\ j = 1,\ 2,\ ...,\ N$, e as restrições de conjunto (*set constraints*) são indicadas por um conjunto $S \subset \mathbb{R}^N$. As soluções que obedecem essas restrições são chamadas de soluções válidas ou viáveis e o conjunto de todas as soluções válidas $\mathbb{S}$ é chamado de espaço de busca (*search space*). Assim, um problema de otimização é indicado pela Equação 1.1 (LUENBERGER; YE, 2016, p. 1)

$$\begin{aligned} &maximizar\ f(x) \\ &\quad sujeito\ a\ g_i(x) = 0,\ i = 1,\ 2,\ ...,\ M \\ &\qquad\qquad h_j(x) \leq 0,\ j = 1,\ 2,\ ...,\ N \\ &\quad e\ x \in S \end{aligned} \tag{1.1}$$

Os problemas de otimização são divididos em lineares e não-lineares. Os problemas onde a função objetiva e as equações e inequações de restrição são lineares são chamados de problemas lineares. De acordo com o Teorema Fundamental da Programação Linear, a solução de um problema linear é um ponto extremo ou vértice do polítopo formado pelas equações e inequações de restrição do problema. Assim, os problemas lineares podem ser considerados problemas de busca em grafos. Ou seja, todos os problemas lineares podem ser resolvidos na Programação Matemática e a solução é conhecida (LUENBERGER; YE, 2016, p. 11).

Em contrapartida, todos os problemas que não são completamente lineares são chamados de problemas não-lineares. Ao contrário dos lineares, os problemas não-lineares podem ter várias soluções distintas, chamadas de máximos ou mínimos relativos ou locais. O ponto x^* é chamado de máximo relativo ou local de f em S, se há um $\epsilon > 0$ tal que $f(x^*) \geq f(x)$ para todo $x \in S$ a uma distância ϵ de x^*, ou seja: $|x - x^*| \leq \epsilon$. Adicional-

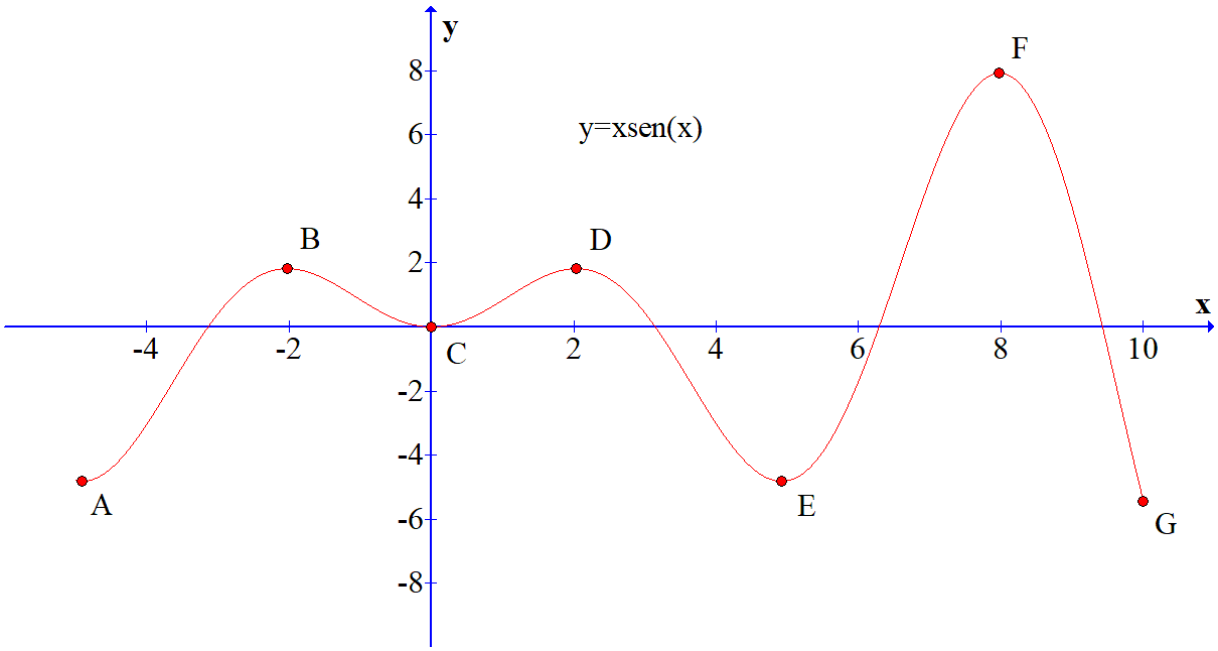

Figura 1 – Ótimos da função $y = xsen(x)$ no intervalo $[-5,\ +10]$.

mente, o ponto x^* é chamado de máximo absoluto ou global de f em S, se $f(x^*) \geq f(x)$ para todo $x \in S$. As definições do mínimo relativo ou local e do mínimo absoluto ou global são análogas. Os problemas que só têm um máximo ou mínimo são chamados de unimodais e os que têm vários, de multimodais (LUENBERGER; YE, 2016, p. 179).

Os máximos e mínimos relativos e globais da função $y = xsen(x)$ no intervalo $[-5,\ +10]$ são mostrados na Figura 1. Os pontos A, C e E são mínimos relativos dessa função, e os pontos B e D são máximos relativos. O ponto F é o máximo global e o ponto G é o mínimo global.

1.2 Algoritmos de Otimização Global

Os algoritmos usados para resolver os problemas não-lineares são divididos em clássicos e modernos. Quase todos os algoritmos clássicos, que também são chamados de matemáticos ou determinísticos, dependem da continuidade das derivadas da função objetiva. Ou seja, eles não podem ser usados se essas derivadas não são contínuas no espaço de busca ou não são conhecidas. Adicionalmente, todos os algoritmos clássicos dependem da continuidade da função objetiva. Ou seja, eles não podem ser usados se essa função não é contínua no espaço de busca ou não é uma função matemática. Além disso, os algoritmos clássicos não procuram o ótimo global do problema, mas sim um ótimo relativo.

Essas limitações provocaram o desenvolvimento dos algoritmos de otimização global, que também são chamados de algoritmos de otimização modernos, porque são mais novos do que os clássicos; probabilísticos ou estocásticos, porque usam variáveis aleatórias; ou

de meta-heurísticas, porque são heurísticas baseadas em heurísticas. Ao contrário dos clássicos, os algoritmos de otimização global não dependem nem da continuidade das derivadas da função objetiva, nem de uma formulação matemática dela. De fato, esses algoritmos podem ser usados até com uma simulação do problema ou medida real. Além disso, eles podem achar o ótimo global do problema, embora não garantam esse resultado.

Os algoritmos de otimização global são baseados em heurísticas. A palavra "heurística" é derivada da palavra "$\epsilon\upsilon\rho\iota\sigma\kappa\omega$" (*"heurisko"*), do grego, que significa: "1. descobrir, encontrar por acaso, encontrar-se com; [...] 2. achar pela averiguação, reflexão, exame, escrutínio, observação, descobrir pela prática e experiência; [...] 3. descobrir por si mesmo, adquirir, conseguir, obter, procurar." (STRONG, 2002, G2147) Nesse contexto, as heurísticas são técnicas probabilísticas para achar a solução de um problema. Ou seja, técnicas que podem achar uma solução muito boa, mas não garantem que vão achar essa solução, nem a qualidade da solução que vão achar.

As heurísticas são usadas quando as outras técnicas não são viáveis, ora porque as derivadas da função objetiva não são contínuas no espaço de busca ou não são conhecidas, ora porque a função objetiva não é contínua no espaço de busca ou não é uma função matemática, ora porque ela tem muitos ótimos relativos. Nesse caso, a única alternativa é analisar todas as soluções do problema para selecionar a melhor, fazendo uma busca exaustiva. No entanto, a complexidade dessa técnica é exponencial, de acordo com o número de variáveis do problema, enquanto a da maioria das heurísticas é constante. Assim, a confiabilidade de uma busca exaustiva pode ser trocada pela rapidez de uma heurística (WEISE, 2009, p. 21).

Quase todas as heurísticas são baseadas em fenômenos da natureza, por exemplo: o Algoritmo Genético, baseado na Teoria da Evolução; o Enxame de Partículas, baseado nos enxames de insetos; e, claro, o Algoritmo de Busca Gravitacional, baseado na Lei da Gravidade; entre outras. No entanto, algumas são baseadas em outras ideias, por exemplo: Otimização Aleatória, Escalada da Colina, etc. Essa variedade de meta-heurísticas é explicada pelos teoremas de "não existe almoço grátis" (*no free lunch theorems*), que dizem que o ganho na qualidade da solução de uma classe de problemas, de uma meta-heurística, significa a perda na qualidade da solução de outra classe de problemas.

De fato, algumas meta-heurísticas são boas em alguns problemas e outras, em outros problemas: algumas são mais apropriadas em problemas discretos e outras, em problemas contínuos; algumas são mais apropriadas em problemas unimodais e outras, em problemas multimodais; etc. Não há uma meta-heurística melhor do que todas as outras em todas as classes de problemas (WOLPERT; MACREADY, 1997).

1.3 Algoritmo de Busca Gravitacional

O Algoritmo de Busca Gravitacional — *Gravitational Search Algorithm (GSA)* — é um algoritmo de otimização global baseado nas leis da gravidade e da dinâmica de Newton (RASHEDI et al., 2009). Nesse algoritmo, as soluções candidatas são representadas por partículas, que se atraem umas às outras, de acordo com a Lei da Gravidade; e se movimentam, de acordo com as leis da dinâmica. A qualidade de determinada solução é medida pela massa da partícula, que é atualizada em cada iteração. Assim, as partículas avançam na direção das melhores soluções conhecidas, de uma forma geral.

Introduzido em 2009, o *GSA* já tem várias versões, por exemplo: *Binary Gravitational Search Algorithm*, *Fuzzy Gravitational Search Algorithm* e *Black Hole Gravitational Search Algorithm*, etc.; e se destaca em várias aplicações, por exemplo: despacho econômico, modelagem de filtros digitais, redes neuronais, etc.; que são mostradas no Capítulo 3. Ou seja, a generalidade e a confiabilidade do *GSA* já foram demonstradas.

No entanto, a convergência do *GSA* depende de alguns parâmetros — número de partículas, constante gravitacional inicial, função gravitacional, etc. — que são determinados empiricamente (SABRI et al., 2013). A maioria das aplicações do *GSA* usa uma constante gravitacional inicial determinada por tentativas e correções, porque uma constante apropriada para uma aplicação não é necessariamente apropriada para outra, e não há uma fórmula para determinar a constante mais apropriada para cada aplicação. Assim, a generalidade do algoritmo é comprometida.

Neste trabalho, é proposta uma heurística para determinar uma constante gravitacional inicial do *GSA*, que seja proporcional ao espaço de busca do problema: a Constante Gravitacional Normalizada — *Normalized Gravitational Constant (NGC)*. Essa heurística é baseada na Teoria de Brans-Dicke, que diz que a força gravitacional é diretamente proporcional ao raio do universo visível e inversamente proporcional à massa; e justificada pela observação de que a constante gravitacional inicial deve ser proporcional ao espaço de busca.

1.4 Objetivos

Os objetivos deste trabalho são:

1. apresentar uma heurística para determinar uma constante gravitacional inicial do *GSA*, proporcional ao espaço de busca do problema;

2. confirmar a observação de que a constante gravitacional inicial do *GSA* deve ser proporcional ao espaço de busca do problema, justificando a heurística apresentada; e

3. confirmar que a heurística apresentada é apropriada para problemas unimodais e multimodais, com espaços de busca variados.

1.5 Organização

Este trabalho está organizado em seis capítulos. No Capítulo 2, é feita uma revisão dos problemas de otimização e dos algoritmos de otimização global. No Capítulo 3, é feita uma revisão de algumas versões e aplicações do *GSA* da literatura. No Capítulo 4, a *Normalized Gravitational Constant* é apresentada e justificada. No Capítulo 5, os experimentos realizados são mostrados e os resultados são analisados. No Capítulo 6, é feita uma revisão da heurística proposta e dos resultados dos experimentos, e algumas possibilidades de pesquisa são apontadas.

2 Revisão da teoria

Neste capítulo, é feita uma revisão sobre os problemas e algoritmos de otimização. Na seção 2.1, algumas definições e variações dos problemas de otimização são mostradas. Na seção 2.2, alguns algoritmos de otimização global são mostrados. Na seção 2.3, o Algoritmo de Busca Gravitacional é mostrado.

2.1 Problemas de Otimização

Conforme dito no Capítulo 1, os problemas de otimização têm uma função objetiva e podem ter um conjunto de equações e inequações de restrição ou não. No entanto, alguns problemas de otimização têm várias funções objetivas distintas, e uma solução ótima para uma função pode não ser ótima para outra. Esses problemas são tratados na subseção 2.1.1. Além disso, a maioria dos algoritmos de otimização não considera as restrições funcionais, que são tratadas na subseção 2.1.2.

2.1.1 Problemas Multi-Objetivos

Os problemas de otimização podem ser divididos em uni-objetivos (*single-objective*) e multi-objetivos (*multi-objective*). Os problemas uni-objetivos só têm um objetivo, representado por uma função objetiva. Em contrapartida, os multi-objetivos têm vários objetivos, representados por várias funções objetivas: $f_i : \mathbb{R}^M \to \mathbb{R},\ i = 1,\ 2,\ ...,\ N$, ou uma função multi-objetiva: $f : \mathbb{R}^M \to \mathbb{R}^N$, onde M é o número de variáveis do problema e N é o número de objetivos. Nesse caso, uma solução ótima para uma função pode não ser ótima para outra. As técnicas mais usadas para resolver os problemas multi-objetivos são a Somatória Ponderada e a Otimização de Pareto.

Somatoria Ponderada

A Somatória Ponderada é a técnica mais simples para resolver um problema multi-objetivo. Nessa técnica, as funções objetivas são combinadas em uma função auxiliar $f'(x)$, igual à somatória de todas as funções objetivas, cada qual multiplicada por um peso, de acordo com a Equação 2.1

$$f'(x) = \sum_{i=1}^{N} (w_i f_i(x)) \tag{2.1}$$

onde w_i é o peso da função objetiva f_i e N é o número de objetivos do problema.

Essa função é considerada a função objetiva do problema e, assim, um problema multi-objetivo é transformado em um problema uni-objetivo. No entanto, ela não é apropriada

para funções objetivas que têm notações assintóticas diferentes umas das outras. Além disso, já que a notação assintótica de uma função depende da modelagem matemática, essa técnica também não é apropriada para funções objetivas que não têm uma modelagem matemática (WEISE, 2009, p. 29).

Otimização de Pareto

A Otimização de Pareto é a técnica mais usada para resolver um problema multi-objetivo. Nessa técnica, a solução x_1 domina — ou seja, é considerada melhor do que — a solução x_2, se a solução x_1 é melhor do que a solução x_2 em uma função objetiva e não é pior nas outras, de acordo com a Equação 2.2

$$\begin{aligned} x_1 \vdash x_2 \Leftrightarrow & \forall i \in [1,\ N] : w_i f_i(x_1) \geq w_i f_i(x_2) \wedge \\ & \exists j \in [1,\ N] : w_j f_j(x_1) > w_j f_j(x_2) \end{aligned} \tag{2.2}$$

onde f_i é a i-ésima função objetiva do problema, o peso w_i é determinado pela Equação 2.3 e N é o número de objetivos do problema.

$$w_i = \begin{cases} 1 & \text{se a função } f_i \text{ deve ser maximizada} \\ -1 & \text{se a função } f_i \text{ deve ser minimizada} \end{cases} \tag{2.3}$$

onde f_i é a i-ésima função objetiva do problema.

Considerando essa definição, a solução x^* é ótima se ela não é dominada em nenhuma função objetiva, de acordo com a Equação 2.4

$$x^* \in X^* \Leftrightarrow \nexists x \in \mathbb{S} : x \vdash x^* \tag{2.4}$$

onde X^* é a Fronteira de Pareto *(Pareto Front)*, ou seja, o conjunto das soluções consideradas ótimas, de acordo com essa técnica (WEISE, 2009, p. 31).

2.1.2 Problemas com Restrições

Os problemas de otimização também podem ser divididos em problemas com restrições (*constrained problems*) e problemas sem restrições (*unconstrained problems*). Apesar dessa definição, só as restrições de conjunto são consideradas na maioria dos algoritmos de otimização global. As restrições funcionais são tratadas por técnicas próprias, que podem ser combinadas com qualquer algoritmo. As mais usadas são a Pena de Morte, as Funções de Penalidade, as Restrições como Objetivos Adicionais e o Método das Inequações.

Pena de Morte

A Pena de Morte (*Death Penalty*) é a técnica mais simples para tratar as restrições. Nessa técnica, as soluções que violam alguma restrição são simplesmente ignoradas e

não são consideradas na otimização. Assim, as informações dessas soluções também são ignoradas.

Funções de Penalidade

Uma das técnicas mais populares, principalmente nas funções uni-objetivas, são as Funções de Penalidade (*Penalty Functions*). Nessa técnica, a função objetiva é combinada com as equações e inequações de restrição do problema em uma função auxiliar $f'(x)$, de acordo com a Equação 2.5

$$f'(x) = F(f(x),\ g(x),\ h(x)) \tag{2.5}$$

onde $f(x)$ é a função objetiva, $g(x)$ são as equações de restrição, $h(x)$ são as inequações de restrição e F é a função de penalidade.

Essa combinação deve ser feita de uma forma tal que o valor de uma solução viável dessa função seja maior do que o de uma inviável. Essa função é considerada a função objetiva do problema e, assim, um problema com restrições é transformado em um problema sem restrições.

Restrições como Objetivos Adicionais

As restrições de um problema também podem ser modeladas como objetivos adicionais. Por exemplo: uma equação de restrição $g(x) = 0$ pode ser modelada como um problema de minimização, por exemplo: $f'(x) = |g(x)|$, e uma inequação $h(x) \geq 0$ pode ser modelada como um problema de maximização, por exemplo: $f''(x) = h(x)$. Assim, um problema com restrições é transformado em um problema sem restrições.

Método das Inequações

No Método das Inequações (*Method of Inequalities*), um intervalo de interesse $[\check{r}_i,\ \hat{r}_i]$ é definido para cada função objetiva, e as soluções são divididas em três conjuntos:

1. A solução cumpre todos os objetivos, de acordo com a Equação 2.6

$$\check{r}_i \leq f_i(x) \leq \hat{r}_i,\ \forall i \in [1,\ N] \tag{2.6}$$

2. A solução cumpre alguns objetivos, mas não todos, de acordo com a Equação 2.7

$$\begin{gathered}(\exists i \in [1,\ N] :\ \check{r}_i \leq f_i(x) \leq \hat{r}_i) \wedge \\ (\exists j \in [1,\ N] :\ f_j(x) < \check{r}_j\ \vee\ f_j(x) > \hat{r}_j)\end{gathered} \tag{2.7}$$

3. A solução não cumpre objetivo algum, de acordo com a Equação 2.8

$$f_i(x) < \check{r}_i \ \vee \ f_i(x) > \hat{r}_i, \ \forall i \in [1, \ N] \tag{2.8}$$

As soluções são comparadas de acordo com essa classificação, seguindo três regras:

1. As soluções que cumprem todos os objetivos são melhores do que as outras.
2. As soluções que não cumprem objetivo algum são piores do que as outras.
3. Uma solução só pode ser comparada com outra do mesmo conjunto.

Assim, elas avançam na direção da Fronteira de Pareto (WEISE, 2009, p. 33).

2.2 Algoritmos de Otimização Global

Quase todos os algoritmos clássicos são diretos, ou seja, a solução do problema é determinada em uma quantidade finita de operações. Em contrapartida, todos os algoritmos de otimização global são iterativos, ou seja, a solução do problema não é determinada em uma quantidade finita de operações, mas sim aproximada em cada iteração. Esses algoritmos passam por uma série de soluções: x_1, x_2, ..., x_N, cada qual em uma iteração do algoritmo. Além disso, a solução da iteração k é baseada na solução da iteração $k-1$ e melhor do que ela ou igual. A característica iterativa é importante, por causa da variedade dos problemas de otimização e da generalidade dos algoritmos. Um algoritmo iterativo é mostrado no Algoritmo 1.

Algoritmo 1 Algoritmo Iterativo

```
as soluções são inicializadas
iteração ← 1
while a condição de parada não é alcançada do
    as soluções são tratadas e atualizadas
    iteração ← iteração + 1
end while
return a melhor solução encontrada
```

As principais características dos algoritmos de otimização global são a exploração local ou intensificação (*exploitation*) e a exploração global ou diversificação (*exploration*). A intensificação é a habilidade de achar uma solução mais próxima de um ótimo conhecido, enquanto a diversificação é a habilidade de achar um ótimo no espaço de busca. Assim, a intensificação é mais importante nos problemas unimodais e a diversificação, nos problemas multimodais. "Como uma regra prática: você gostaria de usar um algoritmo mais

intensificante, porque ele é mais rápido; mas, quanto mais 'feio' é o espaço de busca, mais você vai ser obrigado a usar um algoritmo mais diversificante." [1]

Um dos maiores problemas dos algoritmos de otimização global é a convergência prematura. Quando um algoritmo não consegue achar soluções fora da vizinhança de um ótimo em uma execução — ou seja, quando ele não consegue mais realizar a diversificação — diz-se que ele convergiu. A convergência não é um problema, todos os algoritmos devem convergir; mas quando um algoritmo converge em um ótimo relativo e não consegue achar o global, a convergência é prematura. Várias técnicas para contornar esse problema são conhecidas, mas a convergência prematura não pode ser completamente eliminada; porque não se pode determinar se todos os ótimos de um problema já foram achados, senão usando a busca exaustiva (WEISE, 2009, p. 58).

Os algoritmos de otimização global são divididos em algoritmos de uma solução (*single solution algorithms*) e algoritmos de população (*population-based algorithms*) ou enxame (*swarm-based algorithms*). Os algoritmos de uma solução só fazem uma avaliação da função objetiva por iteração e os algoritmos de população fazem várias. Três algoritmos de uma solução: a Busca Aleatória, a Escalada da Colina e a Escalada da Colina Estocástica, são mostrados nas subseções 2.2.1, 2.2.2 e 2.2.3, respectivamente; e dois algoritmos de população: o Algoritmo Genético e o Enxame de Partículas, são mostrados nas subseções 2.2.4 e 2.2.5, respectivamente.

2.2.1 Busca Aleatória

A Busca Aleatória — *Random Search (RS)* — é o algoritmo de otimização global mais simples que se poderia imaginar. Em cada iteração desse algoritmo, um solução aleatória é selecionada no espaço de busca. A melhor solução é guardada e retornada no final. Esse algoritmo é mostrado no Algoritmo 2 (LUKE, 2009, p. 20).

Algoritmo 2 Busca Aleatória

```
solução principal ← uma solução aleatória do espaço de busca
iteração ← 1
while a condição de parada não é alcançada do
  solução auxiliar ← uma solução aleatória do espaço de busca
  if a solução auxiliar é melhor do que a principal then
    solução principal ← solução auxiliar
  end if
  iteração ← iteração+1
end while
return solução principal
```

[1] "*As a rule of thumb: you'd like to use a highly exploitative algorithm (it's fastest), but the "uglier" the space, the more you will have no choice but to use a more explorative algorithm.*" (LUKE, 2009, p. 20)

2.2.2 Escalada da Colina

A Escalada da Colina — *Hill Climbing (HC)* — é uma meta-heurística baseada no Gradiente Descendente, que não depende das derivadas da função objetiva. Em cada iteração desse algoritmo, uma solução aleatória é selecionada na vizinhança da melhor solução conhecida, que é atualizada em cada iteração e retornada no final. Esse algoritmo é mostrado no Algoritmo 3.

Algoritmo 3 Escalada da Colina

```
solução principal ← uma solução aleatória do espaço de busca
iteração ← 1
while a condição de parada não é alcançada do
  solução auxiliar ← uma solução aleatória na vizinhança da solução principal
  if a solução auxiliar é melhor do que a principal then
    solução principal ← solução auxiliar
  end if
  iteração ← iteração+1
end while
return solução principal
```

Dependendo do tamanho da vizinhança, o *HC* não consegue achar ótimos fora da vizinhança de convergência da solução principal e, portanto, não pode ser considerado um algoritmo de otimização global (WEISE, 2009, p. 253).

2.2.3 Escalada da Colina Estocástica

A Escalada da Colina Estocástica — *Stochastic Hill Climbing (SHC)* — é uma combinação do *HC* e do *RS*. Nessa técnica, várias escaladas da colina são realizadas e a melhor solução é retornada. Esse algoritmo é mostrado no Algoritmo 4.

2.2.4 Algoritmo Genético

Os algoritmos evolutivos — *evolutionary algorithms (EAs)* — são algoritmos de população que usam heurísticas baseadas na Teoria da Evolução — por exemplo: seleção, cruzamento e mutação — para achar a solução de um problema. Os algoritmos evolutivos não são as únicas técnicas da Ciência da Computação baseadas na Biologia, ao contrário, várias técnicas baseadas na Biologia são conhecidas, por exemplo: redes neuronais artificiais (*artificial neural networks*), mapas auto-organizantes (*self-organizing maps*), aprendizado profundo (*deep learning*), etc.

O Algoritmo Genético — *Genetic Algorithm (GA)* — é um algoritmo evolutivo genérico, que foi desenvolvido por Holland e Goldberg (HOLLAND; GOLDBERG, 1988) nas década de 1960 e 1970. Esse algoritmo é tão importante, que alguns estudiosos chamam todos os algoritmos de otimização global de algoritmos evolutivos. Nesse algoritmo, as so-

Algoritmo 4 Escalada da Colina Estocástica

```
solução final ← uma solução aleatória no espaço de busca
iteração ← 1
while a condição de parada 1 não é alcançada do
  solução principal ← uma solução aleatória no espaço de busca
  while a condição de parada 2 não é alcançada do
    solução auxiliar ← uma solução aleatória na vizinhança da principal
    if a solução auxiliar é melhor do que a principal then
      solução principal ← solução auxiliar
    end if
    iteração ← iteração+1
  end while
  if a solução principal é melhor do que a final then
    solução final ← solução principal
  end if
end while
return solução final
```

luções candidatas são representadas por cromossomos, genótipos ou indivíduos, conforme a Figura 2.

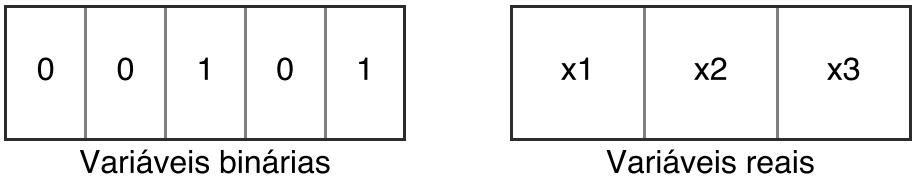

Figura 2 – Um cromossomo com variáveis binárias e um com variáveis reais.

A qualidade de determinada solução é representada pela aptidão (*fitness*) do indivíduo, baseada na função objetiva. Os indivíduos mais aptos são selecionados para o cruzamento, gerando novas soluções candidatas; e os menos aptos são eliminados da população em cada iteração, provocando a evolução (WEISE, 2009, p. 131). Um algoritmo genético é mostrado no Algoritmo 5.

Algoritmo 5 Algoritmo Genético

```
a população do algoritmo é inicializada e avaliada
while a condição de parada não é alcançada do
  alguns indivíduos são selecionados para o cruzamento (seleção)
  novos indivíduos são gerados pela recombinação dos pais (cruzamento)
  esses indivíduos passam por uma mutação
  alguns indivíduos são eliminados da população
end while
return a melhor solução encontrada
```

As principais operações desse algoritmo são a seleção, o cruzamento e a mutação.

Seleção

A operação da seleção é baseada na seleção natural. Nessa operação, alguns indivíduos são selecionados para realizar o cruzamento (GOLDBERG; DEB, 1991). Várias estratégias de seleção são conhecidas, por exemplo:

1. **Seleção Proporcional à Aptidão**

   Na Seleção Proporcional à Aptidão (*Fitness Proportionate Selection*), a probabilidade de que um indivíduo seja selecionado para o cruzamento $p(i)$ é proporcional à aptidão dele, de acordo com a Equação 2.9

   $$p(i) = \frac{q(i)}{\sum_{j=1}^{N} q(j)} \tag{2.9}$$

   onde $q(i)$ é a qualidade da solução i, ou seja, a aptidão do indivíduo i, e N é o número de indivíduos da população.

2. **Seleção por Classifcação**

   Na Seleção por Classificação (*Rank Selection*), todos os indivíduos são colocados em uma lista ordenada pela aptidão e a probabilidade de que um indivíduo seja selecionado para o cruzamento é baseada nessa ordem. Ou seja, se o indivíduo 1 é mais apto do que o 2, então a chance do indivíduo 1 é maior do que a do 2. No entanto, essas chances não variam ainda que o indivíduo 1 seja dez vezes mais apto do que o 2 ou só duas vezes, por exemplo.

3. **Seleção por Torneio**

   Na Seleção por Torneio (*Tournament Selection*), K indivíduos aleatórios são selecionados para um torneio e o melhor é selecionado para o cruzamento. Se $K = 1$, então um indivíduo aleatório é selecionado para o torneio e, portanto, para o cruzamento (Seleção Aleatória). Se $K = N$, então todos os indivíduos são selecionados para o torneio e, portanto, o mais apto é selecionado para o cruzamento (Seleção Truncada).

Cruzamento

A operação do cruzamento (*crossover*) ou da recombinação (*recombination*) é baseada no cruzamento das espécies sexuadas. Nessa operação, os indivíduos selecionados são recombinados e dois indivíduos são gerados, ou seja, duas novas soluções candidatas são geradas (WEISE, 2009, p. 148). Várias estratégias de cruzamento são conhecidas, por exemplo:

1. **Cruzamento de Um Ponto**

No Cruzamento de Um Ponto, um ponto aleatório do cromossomo é selecionado e os genes da esquerda desse ponto vão para um filho e os genes da direita vão para o outro. Um Cruzamento de Um Ponto é mostrado na Figura 3.

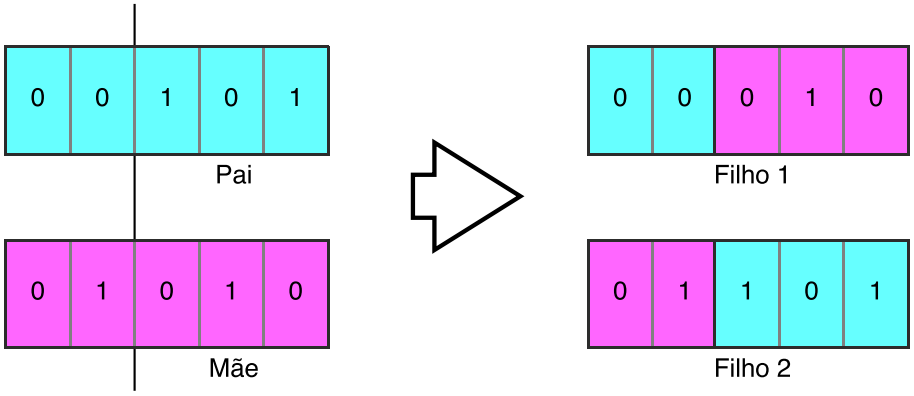

Figura 3 – Cruzamento de Um Ponto

2. **Cruzamento de K Pontos**

   No Cruzamento de K Pontos, K pontos aleatórios do cromossomo são selecionados e os genes de cada divisão vão para um filho alternadamente. Um Cruzamento de Dois Pontos é mostrado na Figura 4.

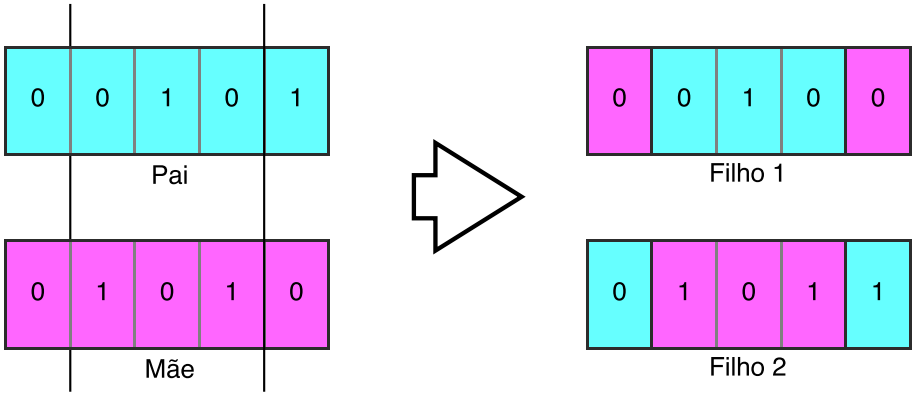

Figura 4 – Cruzamento de Dois Pontos

3. **Cruzamento Uniforme**

   No Cruzamento Uniforme, cada gene tem uma probabilidade de que seja trocado. Um Cruzamento Uniforme é mostrado na Figura 5.

Mutação

Nessa operação, os indivíduos gerados no cruzamento passam por uma mutação, ou seja, as soluções são modificadas (WEISE, 2009, p. 147). Várias estratégias de mutação são conhecidas, por exemplo:

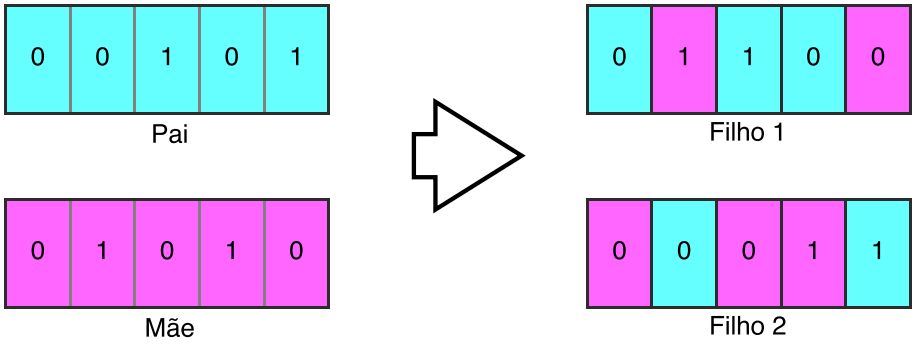

Figura 5 – Cruzamento Uniforme

1. **Mutação de Um Gene**

   Na Mutação de Um Gene, um gene do cromossomo é selecionado e trocado, de acordo com a Figura 6.

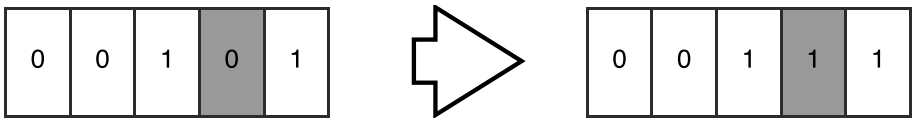

Figura 6 – Mutação de Um Gene

2. **Mutação de K Genes**

   Na Mutação de K Genes, K genes do cromossomo são selecionados e trocados. Uma Mutação de Três Genes é mostrada na Figura 7.

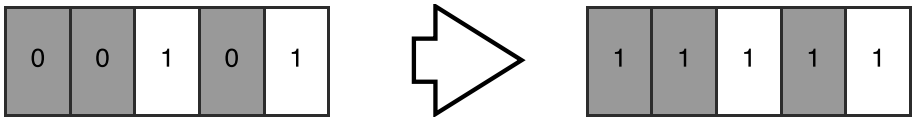

Figura 7 – Mutação de Três Genes

3. **Mutação Uniforme**

   Na Mutação Uniforme, cada gene tem uma probabilidade de que seja trocado. Assim, a quantidade de genes que são trocados não é definida.

2.2.5 Enxame de Partículas

A Otimização do Enxame de Partículas — *Particle Swam Optimization (PSO)* — é um algoritmo de população que foi desenvolvido por Kennedy e Eberhart (KENNEDY; EBERHART, 1995), em 1995. Esse algoritmo é baseado no comportamento de um enxame de insetos procurando comida. Todos os indivíduos se movimentam juntamente na direção de uma porção de comida, mas cada indivíduo também se movimenta sozinho, procurando

outras porções. Quando um indivíduo acha uma porção maior do que a outra, todos vão na direção dela.

Nesse algoritmo, cada solução é representada por um indivíduo. Cada indivíduo se movimenta na direção da maior porção de comida conhecida do enxame, que é a melhor solução do enxame; mas também na direção da maior porção de comida que só ele conhecia, que é a melhor solução do indivíduo. Essas soluções são atualizadas em cada iteração (KENNEDY; EBERHART, 1995). Assim, a velocidade de cada indivíduo tem uma parte dependente do enxame e uma parte independente do enxame, além de uma parte inercial, de acordo com a Equação 2.10

$$\begin{aligned} v_i(t) =& u(N)c_1(sS - x_i(t-1)) + \\ & u(N)c_2(iS_i - x_i(t-1)) + \\ & w(t)v_i(t-1) \end{aligned} \tag{2.10}$$

onde $x_i(t)$ é a posição do indivíduo i na iteração t e $v_i(t)$ é a velocidade do indivíduo i na iteração t, sS é a melhor solução do enxame e iS_i é a melhor solução do indivíduo i, c_1 é a constante do movimento dependente do enxame e c_2 é a constante do movimento independente, $w(t)$ é o peso da inércia na iteração t e $u(N)$ é um vetor de N variáveis aleatórias no intervalo $[0,\ 1]$, e N é o número variáveis do problema.

A posição de cada indivíduo é atualizada em cada iteração, de acordo com a Equação 2.11

$$x_i(t) = x_i(t-1) + v_i(t) \tag{2.11}$$

onde $x_i(t)$ é a posição do indivíduo i na iteração t e $v_i(t)$ é a velocidade do indivíduo i na iteração t.

Um enxame de partículas é mostrado no Algoritmo 6.

A intensificação e a diversificação do *PSO* são controladas pelas constantes do movimento dependente do enxame c_1 e do movimento independente c_2. Quando a constante c_1 é maior do que a constante c_2, todas as partículas se concentram na melhor solução do enxame e realizam a intensificação; mas quando a constante c_2 é maior do que a constante c_1, cada partícula se concentra em uma solução e realiza a diversificação. Essas constantes também podem ser variáveis em função da iteração, de uma forma tal que a diversificação seja maior nas primeiras iterações e a intensificação seja maior nas últimas.

Algoritmo 6 Enxame de Partículas

```
for i = 1 to P do
  x_i(0) ← posição aleatória
  v_i(0) ← velocidade aleatória
end for
t ← 1
while a condição de parada não é alcançada do
  for i = 1 to P do
    v_i(t) ←u(N)c_1(sS − x_i(t − 1))+
             u(N)c_2(iS_i − x_i(t − 1))+
             w(t)v_i(t − 1)
    x_i(t) ← x_i(t − 1) + v_i(t)
    {q(S): a qualidade da solução S}
    if q(x_i(t)) > q(iS_i) then
      iS_i ← x_i(t)
      if q(iS_i) > q(sS) then
        sS ← iS_i
      end if
    end if
  end for
  t ← t + 1
end while
return sS
```

2.3 Algoritmo de Busca Gravitacional

O *GSA* é uma meta-heurística de população baseada nas leis da gravidade e da dinâmica de Newton. As partículas massivas são atraídas umas às outras por uma força chamada de gravidade, e a magnitude dessa força F é determinada pela Lei da Gravidade Equação 2.12

$$F = G\frac{M_1 M_2}{R^2} \tag{2.12}$$

onde G é a constante gravitacional, M_1 é a massa de uma partícula e M_2 é a massa da outra, e R é a distância entre elas.

De acordo com a Física Teórica, a massa de uma partícula pode ser dividida em três:

1. A Massa Gravitacional Ativa — *Active Gravitational Mass (AGM)* — é a medida da força gravitacional que uma partícula faz sobre as outras.

2. A Massa Gravitacional Passiva — *Passive Gravitational Mass (PGM)* — é a medida da reação de uma partícula à força gravitacional.

3. A Massa Inercial *(Inertial Mass)* é a resistência de uma partícula à aceleração.

Além disso, a constante gravitacional não é uma constante de fato, mas sim uma função decrescente do tempo (MANSOURI et al., 1999), de acordo com a Equação 2.13

$$G(t) = G_0 \left(\frac{t}{t_0}\right)^{\alpha}, \quad \alpha < 1 \tag{2.13}$$

onde $G(t)$ é a constante gravitacional no instante t e G_0 é a constante gravitacional no instante t_0.

Assim, a magnitude da força gravitacional F_{12} da partícula 2 sobre a partícula 1 é determinada pela Equação 2.14

$$F_{12} = G(t)\frac{M_2' M_1''}{R_{12}^2} \tag{2.14}$$

onde M_2' é a massa gravitacional ativa da partícula 2, M_1'' é a massa gravitacional passiva da partícula 1 e R_{12} é a distância entre a partícula 1 e a partícula 2.

E a magnitude da aceleração a de uma partícula por uma força é dada pela Equação 2.15

$$a = \frac{F}{M} \tag{2.15}$$

onde F é a força total sobre uma partícula e M é a massa inercial.

2.3.1 Formulação do *GSA*

Nessa técnica, cada solução candidata é representada por uma partícula e cada variável do problema é representada por uma coordenada da posição, de acordo com a Equação 2.16

$$x_i(t) = (x_{i1}(t), x_{i2}(t), ..., x_{iD}(t)), \quad i = 1,\ 2,\ ...,\ P \tag{2.16}$$

onde $x_i(t)$ é a solução i — ou seja, a posição da partícula i — na iteração t, $x_{ij}(t)$, $j = 1,\ 2,\ ...,\ D$, a variável j da solução i — ou seja, a dimensão j da posição da partícula i — na iteração t, P é o número de soluções candidatas do algoritmo (ou partículas) e D é o número de variáveis do problema (ou dimensões do espaço de busca).

A posição de cada partícula é inicializada aleatoriamente no espaço de busca, de acordo com a Equação 2.17

$$x_i(0) = u(D) \circ (L^+ - L^-) + L^- \tag{2.17}$$

onde $u(D)$ é um vetor de D variáveis aleatórias no intervalo $[0,\ 1]$ e $maxLimits$ e $minLimits$ são vetores, onde L_j^+ e L_j^- são os limites máximo e mínimo da dimensão j do espaço de busca, respectivamente, e D é o número de dimensões do problema.

Essas partículas se atraem umas às outras, de acordo com a Lei da Gravidade; e se movimentam, de acordo com as leis da dinâmica. Quando uma partícula se movimenta,

de fato, ela vira uma nova solução candidata, conforme a nova posição. A qualidade de determinada solução é medida pela massa da partícula, que é atualizada em cada iteração. Cada iteração dessa técnica é realizada em três fases:

1. A massa de cada partícula é calculada em três fases:

   a) O valor da função objetiva de cada solução é calculado, de acordo com a Equação 2.18

   $$o_i(t) = f(x_i(t)) \tag{2.18}$$

   onde $o_i(t)$ é o valor da função objetiva da solução i na iteração t e $f(x_i(t))$ é o valor da função objetiva no ponto $x_i(t)$, que é a posição da partícula i na iteração t.

   b) A qualidade de cada solução é calculada, de acordo com a Equação 2.19

   $$q_i(t) = \frac{o_i(t) - w(t)}{b(t) - w(t)} \tag{2.19}$$

   onde $q_i(t)$ é a qualidade da solução i na iteração t, $o_i(t)$ é o valor da função objetiva da solução i na iteração t, $b(t)$ é o melhor valor da função objetiva na iteração t e $w(t)$ é o pior.

   Assim, a dinâmica do *GSA* não depende se a função objetiva deve ser maximizada ou minimizada, nem dos máximos e mínimos dela.

   c) A massa de cada partícula é calculada, de acordo com a Equação 2.20

   $$m_i(t) = \frac{q_i(t)}{\sum_{j=1}^{P} q_j(t)} \tag{2.20}$$

   onde $m_i(t)$ é a massa da partícula i na iteração t, $q_i(t)$ é a qualidade da solução i na iteração t e P é o número de partículas do algoritmo.

   Assim, as maiores massas são dadas às melhores soluções em cada iteração. Assim, a dinâmica do *GSA* não depende do número de partículas do algoritmo, porque a somatória das massas de todas as partículas é igual a 1, independentemente desse número.

2. A força total sobre cada partícula também é calculada em três fases:

   a) A constante gravitacional é uma função decrescente proporcional à constante gravitacional inicial, de acordo com a Equação 2.21

   $$G(t) = G_0 e^{-\alpha t/T} \tag{2.21}$$

onde $G(t)$ é a constante gravitacional na iteração t, G_0 é a constante gravitacional inicial, a constante α determina a magnitude da variação de $G(t)$ e T é o número máximo de iterações do algoritmo.

b) Considerando que a massa gravitacional ativa de cada partícula é dada pela Equação 2.20 e a massa gravitacional passiva é igual a 1, a força da partícula j sobre a partícula i é calculada, de acordo com a Equação 2.22

$$F_{ij}(t) = G(t)\frac{(x_j(t) - x_i(t))}{|x_j(t) - x_i(t)|}m_j(t) \tag{2.22}$$

onde $F_{ij}(t)$ é a força da partícula j sobre a partícula i na iteração t, $x_i(t)$ é a posição da partícula i na iteração t, $x_j(t)$ é a posição da partícula j na iteração t, $m_j(t)$ é a massa gravitacional ativa da partícula j na iteração t e $G(t)$ é a constante gravitacional na iteração t.

Assim, a força da partícula j sobre a partícula i não depende da distância entre elas, senão da direção, e a Equação 2.22 é simplificada para a Equação 2.23

$$F_{ij}(t) = G(t) < x_j(t) - x_i(t) > m_j(t) \tag{2.23}$$

onde $< x_j(t) - x_i(t) >$ é um versor (vetor unitário) na direção da partícula j até a partícula i.

c) A força total sobre cada partícula é calculada, de acordo com a Equação 2.24

$$F_i(t) = \sum_{j=1}^{P}(u(D) \circ F_{ij}(t)), \quad j \neq i \tag{2.24}$$

onde $F_i(t)$ é a força total sobre a partícula i na iteração t, $F_{ij}(t)$ é a força da partícula j sobre a partícula i na iteração t, $u(D)$ é um vetor de D variáveis aleatórias no intervalo $[0,\ 1]$, P é o número de partículas do algoritmo e D é o número de dimensões do problema.

Combinando a Equação 2.23 com a Equação 2.24, é obtida a Equação 2.25

$$F_i(t) = G(t)\sum_{j=1}^{P}(u(D) \circ < x_j(t) - x_i(t) > m_j(t)), \quad j \neq i \tag{2.25}$$

onde se observa que a força total sobre a partícula i é proporcional à constante gravitacional.

3. Finalmente, a posição de cada partícula é atualizada em duas fases:

a) Considerando que a massa inercial de cada partícula é igual a 1 e a aceleração de cada partícula é igual à força total sobre ela, a velocidade de cada partícula é atualizada, de acordo com a Equação 2.26

$$v_i(t) = u(D) \circ v_i(t-1) + F_i(t) \tag{2.26}$$

onde $v_i(t)$ é a velocidade da partícula i na iteração t, $F_i(t)$ é a força total sobre a partícula i na iteração t e $u(D)$ é um vetor de D variáveis aleatórias no intervalo $[0,\ 1]$, D é o número de dimensões do problema e $v_i(0) = 0$.

b) A posição de cada partícula é atualizada, de acordo com a Equação 2.27

$$x_i(t) = x_i(t-1) + v_i(t) \tag{2.27}$$

onde $x_i(t)$ é a posição da partícula i na iteração t e $v_i(t)$ é a velocidade da partícula i na iteração t.

Quando uma partícula sai do espaço de busca — ou seja, quando uma partícula realiza uma fuga — ela é reposicionada. Para que essa reposição seja realizada, as dimensões da posição da partícula que passaram dos limites do espaço de busca são novamente inicializadas aleatoriamente. Uma fuga e uma reposição são mostradas na Figura 8. Uma partícula se movimenta da posição $(x_1,\ y_1)$ para a posição $(x_2,\ y_2)$, realizando uma fuga. Então, a dimensão x da posição dessa partícula, que passou do limite do espaço de busca, é novamente inicializada aleatoriamente, em x_3. Assim, o verdadeiro deslocamento dessa partícula é $(x_3,\ y_2) - (x_1,\ y_1)$.

A execução é finalizada quando a condição de parada — que pode ser um número de iterações ou tempo, por exemplo — é alcançada, e então a melhor solução é retornada.

Resumindo, cada solução é representada por uma partícula. Essas partículas se atraem umas às outras e se movimentam, de acordo com a mecânica clássica. A posição e a massa de cada partícula são atualizadas em cada iteração. Assim, as partículas avançam na direção das melhores soluções conhecidas, de uma forma geral. A melhor solução encontrada é retornada quando a condição de parada é alcançada. Um diagrama do *GSA* é mostrado na Figura 9.

2.3.2 Intensificação e Diversificação do *GSA*

A diversificação e a intensificação do *GSA* são determinadas pela movimentação das partículas. Se a velocidade de uma partícula é grande, independentemente da direção, ela realiza a diversificação; senão, ela realiza a intensificação. Assim, a movimentação das

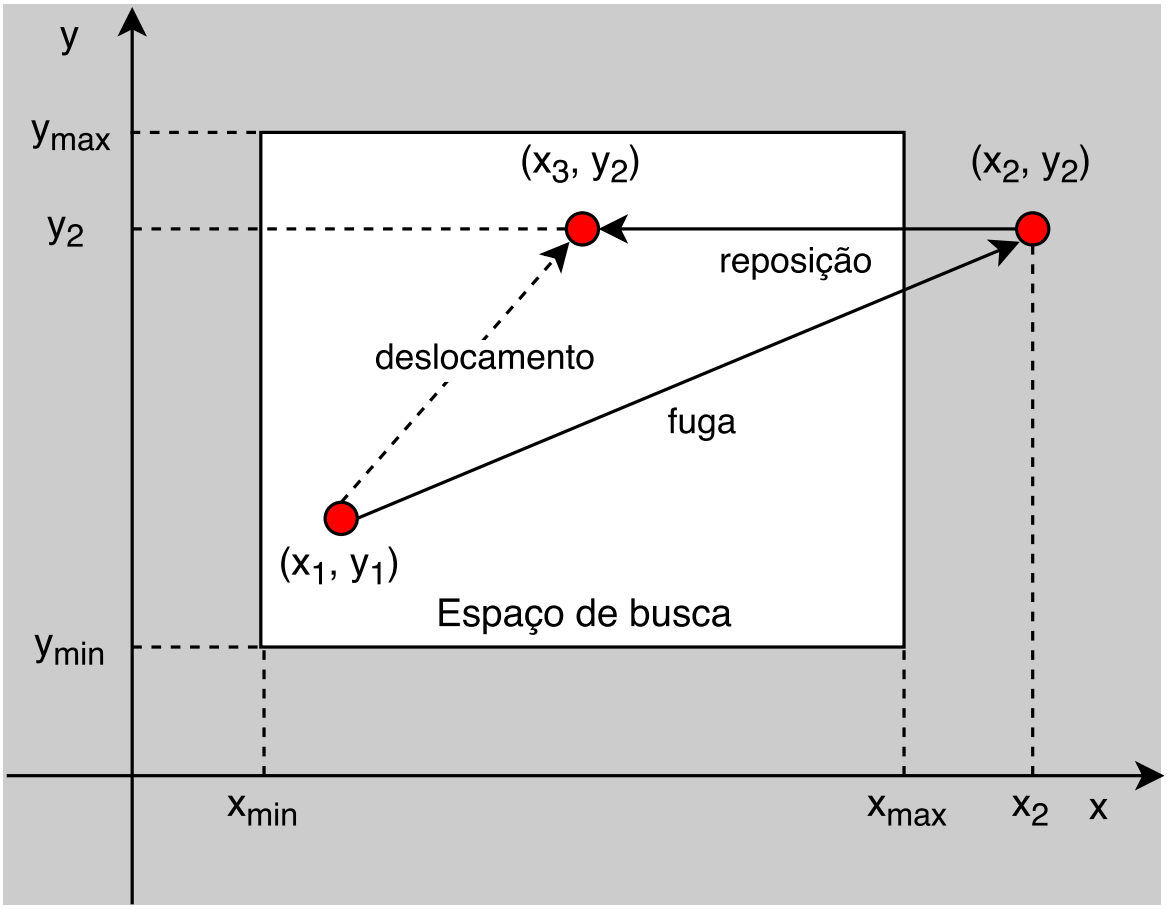

Figura 8 – Uma fuga e a reposição de uma partícula

partículas deve ser grande nas primeiras iterações e pequena nas últimas, para que a diversificação seja grande nas primeiras iterações e pequena nas últimas, e a intensificação seja pequena nas primeiras iterações e grande nas últimas. Considerando que a aceleração de uma partícula é proporcional à constante gravitacional, portanto, a constante gravitacional deve ser uma função decrescente no tempo (RASHEDI et al., 2009).

Além disso, usando a estratégia elitista, só algumas partículas — chamadas de partículas ativas — são consideradas na fórmula da força (Equação 2.24), de acordo com a Equação 2.28

$$F_i(t) = \sum_{j=1}^{K(t)} (u(D) \circ F_{i,best(j)}(t)), \quad best(j) \neq i \tag{2.28}$$

onde $F_i(t)$ é a força total sobre a partícula i na iteração t, $F_{i,best(j)}(t)$ é a força da partícula $best(j)$ sobre a partícula i na iteração t, $best(j)$ é o índice da j-ésima melhor solução, $u(D)$ é um vetor de D variáveis aleatórias no intervalo $[0,\ 1]$, $K(t)$ é o número de partículas ativas na iteração t e D é o número de dimensões do problema.

O número de partículas ativas $K(t)$ deve ser o máximo, igual ao número de partículas

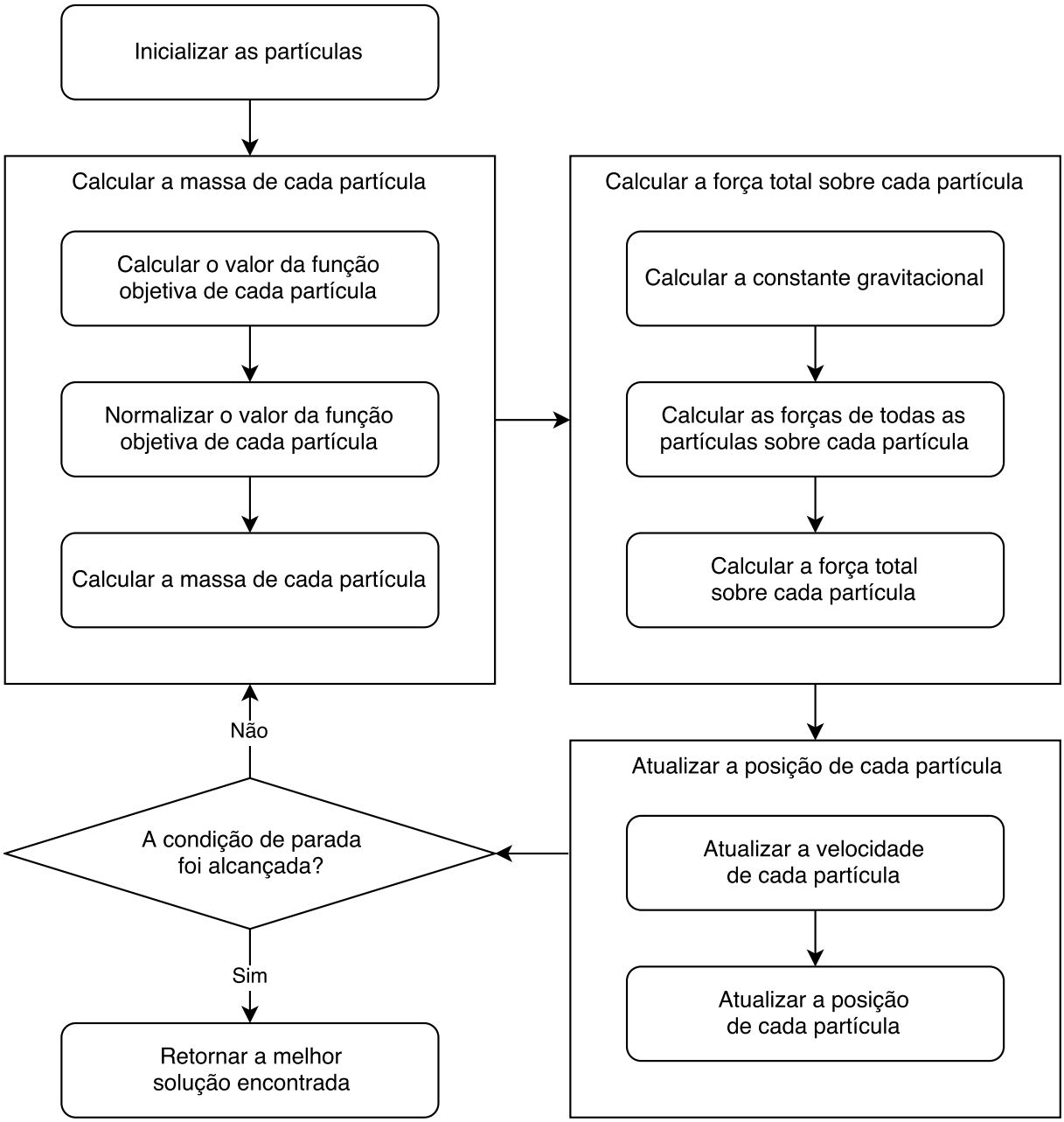

Figura 9 – Diagrama do *GSA*

do algoritmo, na primeira iteração; e diminuir até o mínimo de 1, na última iteração, de acordo com a Equação 2.29

$$K(t) = P - \lfloor (P-1)(t/T) \rceil \tag{2.29}$$

Assim, nas primeiras iterações, quando muitas partículas são ativas, muitas soluções são exploradas, aumentando a diversificação do *GSA*; e, nas últimas, quando poucas partículas são ativas, poucas soluções são exploradas, aumentando a intensificação.

Várias versões do *GSA* foram desenvolvidas nos últimos anos. Algumas dessas versões

são mostradas no Capítulo 3, juntamente com outras publicações sobre esse algoritmo.

3 Revisão Bibliográfica

Neste capítulo, algumas versões e aplicações do *GSA* são mostradas. O principal objetivo dessa revisão é demonstrar que o *GSA* é um algoritmo popular, mas a constante gravitacional inicial ainda não foi devidamente estudada.

3.1 *Binary GSA*

Rashedi et al. apresentaram uma versão binária do *GSA — Binary GSA (BGSA)* (RASHEDI et al., 2010). O BGSA modifica o GSA nos seguintes itens:

1. Todas as variáveis do problema, tanto as discretas quanto as contínuas, são convertidas em um vetor de bits. Por exemplo, se um problema possui duas variáveis, uma real de 32 bits e uma inteira de 8 bits, elas serão convertidas em um vetor de *booleans* de dimensão 40 (32+8). Assim, a posição de cada partícula é representada por um vetor de bits.

2. Na atualização da posição, cada dimensão da posição de uma partícula tem probabilidade s de mudar de 0 para 1 ou vice-versa, de uma forma proporcional à dimensão relativa da velocidade, de acordo com a Equação 3.1

$$s(v_i^d(t)) = |tanh(v_i^d(t))| \tag{3.1}$$

onde $v_i^d(t)$ é a dimensão d da velocidade da partícula i na iteração t.

A atualização da posição é realizada, de acordo com a Equação 3.2

$$x_i^d(t) = \begin{cases} complement(x_i^d(t-1)) & random < s(v_i^d(t)) \\ x_i^d(t-1) & \text{do contrário} \end{cases} \tag{3.2}$$

onde $random$ é uma variável aleatória no intervalo $[0,\ 1]$, $x_i^d(t)$ é a dimensão d da posição da partícula i na iteração t e $complement(x_i^d(t-1))$ é o complemento binário de $x_i^d(t-1)$, ou seja, $complement(0) = 1$ e $complement(1) = 0$.

O *BGSA* foi comparado com o *GA* e com o *Binary PSO (BPSO)*, em experimentos com várias funções de referência discretas e contínuas, e se mostrou melhor do que ambos na maioria das funções.

Foi realizada uma série de experimentos para determinar a constante gravitacional inicial mais apropriada, variando nos valores $G_0 = 50$, 100 e 150. A constante $G_0 = 100$ foi a melhor na maioria dos problemas, mas não em todos. Concluiu-se que essa constante é um parâmetro dependente do problema.

3.2 *Multi-objective GSA*

Hassanzadeh e Rouhani apresentaram uma versão do *GSA* adaptada para problemas multi-objetivos — *Multi-Objective GSA (MO-GSA)* (HASSANZADEH; ROUHANI, 2010). O MO-GSA usa algumas estratégias conhecidas da otimização multi-objetiva, tais como a operação de mutação uniforme, que adiciona um valor aleatório a uma variável aleatória de uma solução; e a política elitista, que guarda as soluções não-dominadas em uma grade.

Duas métricas da otimização multi-objetiva foram consideradas pelos autores, o espaçamento e a distância geracional:

1. O espaçamento (*spacing*) S é a medida da uniformidade da distância entre as soluções do sistema, de acordo com a Equação 3.3

$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(d_i - \overline{d})^2}, \quad i = 1,\ 2,\ ...,\ N \tag{3.3}$$

   onde a variável d_i é dada pela Equação 3.4

$$d_i = min\sum_{k=1}^{O}|f_k(x_i) - f_k(x_j)|, \quad j = 1,\ 2,\ ...,\ N,\ j \neq i \tag{3.4}$$

   onde $f_k(x_i)$ é o valor da função objetiva k da partícula i, e O é o número de funções objetivas do problema, $\overline{d}$ é a média de todas as variáveis d_i e N é o número de partículas do sistema.

2. A distância geracional (*generational distance*) GD é a medida da distância das soluções encontradas até a fronteira de Pareto, de acordo com a Equação 3.5

$$GD = \frac{1}{N}\sqrt{\sum_{i=1}^{N} p_i^2} \tag{3.5}$$

   onde p_i é a distância entre a solução i e a fronteira de Pareto.

O *MO-GSA* foi comparado com o *Simple Multi-Objective PSO (SMOPSO)* e com o *Pareto Archived Evolution Strategy (PAES)*, em experimentos com três funções de referência conhecidas. Os resultados produzidos pelo *MO-GSA*, de acordo com as duas métricas citadas, foram melhores que os produzidos pelos outros dois algoritmos. No entanto, seu custo computacional, em termos de tempo de execução por iteração, é maior.

A constante gravitacional inicial usada nos experimentos $G_0 = 1.5$ foi determinada experimentalmente.

3.3 *Improved GSA*

Sarafrazi et al. apresentaram uma versão melhorada do *GSA — Improved GSA (IGSA)* —, que usa uma operação de rompimento (*disruption*) (SARAFRAZI et al., 2011). O rompimento consiste na separação das partículas umas das outras, de um grupo de partículas unidas pela gravidade, sempre que o grupo se aproxima de uma partícula com massa muito grande. A operação do rompimento acontece quando a partícula i do enxame se aproxima muito da partícula com massa muito grande j, satisfazendo a Equação 3.6

$$\frac{R_{ij}}{R_i^*} < C \tag{3.6}$$

onde R_{ij} é a distância entre a partícula i e a partícula j, R_i^* é a distância entre a partícula i e a melhor solução da iteração, e a variável C usada para controlar a diversificação e a intensificação dessa operação.

Quando uma partícula é rompida, a sua posição é atualizada, de acordo com a Equação 3.7

$$x_i = \begin{cases} random(-0.5,\ +0.5)R_{ij}x_i' & if\ R_i^* \geq 1 \\ \rho * random(-0.5,\ +0.5)x_i' + x_i' & \text{do contrário} \end{cases} \tag{3.7}$$

onde $random(min, max)$ é uma variável aleatória no intervalo $[min, max]$, x_i' é a posição da partícula i antes do rompimento e x_i é a posição da partícula i depois do rompimento. A função da variável ρ não é descrita pelos autores.

O *IGSA* foi comparado com a versão original do *GSA — Standard GSA (SGSA)* —, com o *Real GA (RGA)* e com o *PSO* em experimentos com treze funções de referência conhecidas. Seus resultados foram melhores do que os dos demais algoritmos, na maioria dos experimentos.

A constante gravitacional original $G_0 = 100$ foi usada nos experimentos, sem justificativas.

3.4 *Fuzzy GSA*

Saeidi-Khabisi e Rashedi apresentaram o *Fuzzy Gravitational Search Algorithm (FGSA)* (SAEIDI-KHABISI; RASHEDI, 2012). No *FGSA*, a lógica difusa (*fuzzy logic*) é usada para determinar a constante α da Equação 2.21, para controlar a intensificação e a diversificação do *GSA*, de acordo com duas métricas conhecidas:

1. A diversidade da população (PD) é dada pela Equação 3.8

$$PD = \frac{R_{ave} - R_{min}}{R_{max} - R_{min}} \tag{3.8}$$

Onde, considerando que R_i^* é a distância da solução i até a melhor, R_{ave} é a média de todas as distâncias R_i^*, R_{max} é a maior e R_{min} é a menor, onde $i = 1,\ 2,\ ...,\ P$, e P é o número de partículas do algoritmo.

2. A medida da convergência (CM) das soluções, de uma iteração para outra, é dada pela Equação 3.9

$$CM = \frac{q_{ave}(t-1) - q_{ave}(t)}{q_{ave}(t)} \tag{3.9}$$

onde $q_{ave}(t)$ é a média das qualidades das soluções da iteração t.

O *FGSA* foi comparado com o *SGSA* em experimentos com algumas funções de referência. Os resultados produzidos pelo *FGSA*, em termos de qualidade da solução e de convergência por iteração, são um pouco melhores que os do *SGSA*. No entanto, seu custo computacional, em termos de tempo de execução por iteração, é maior.

A constante gravitacional original $G_0 = 100$ foi usada nos experimentos, sem justificativas.

3.5 *Discrete GSA*

Dowlatshahi et al. também apresentaram outra versão do *GSA* — *Discrete GSA (DGSA)* —, apropriada para problemas discretos (DOWLATSHAHI et al., 2014). No *DGSA*, o movimento de cada partícula, definido no *GSA* (Equação 2.26), foi dividido em dois tipos diferentes de movimento — um independente e outro dependente da posição das partículas. O comprimento do movimento independente (*independent movement length (IML)*) é dado por $u(D) \circ v_i(t-1)$. O comprimento do movimento dependente (*dependent movement length (DML)*) é dado por $F_i(t)$.

O movimento independente da posição das partículas foi trocado por um operador de movimento independente (*independent movement operator (IMO)*). No *IMO*, uma solução candidata procura por uma solução vizinha melhor do que ela, de acordo com o Algoritmo 7

Algoritmo 7 Operação de Movimento Independente do *DGSA*

```
iteração ← 0
IML_i(t) = random * (IML_i(t − 1) + DML_i(t − 1))
while iteração< IML_i(t) do
    solução ← alguma solução vizinha melhor do que ela
    iteração ← iteração+1
end while
return solução
```

O movimento dependente da posição das partículas foi trocado por um operador de movimento dependente (*dependent movement operator (DMO)*). No *DMO*, uma solução candidata desloca-se na direção das outras soluções do sistema, de acordo com o Algoritmo 8, onde K é o número de partículas ativas (ou de soluções do sistema) e DML_{ij} é dada pela Equação 3.10

$$DML_{ij}(t) = \left\lceil random * G(t)\frac{M_j(t)}{R'_{ij}(t)}R_{ij}(t) \right\rceil \tag{3.10}$$

onde $R'_{ij}(t)$ é uma normalização de $R_{ij}(t)$ no intervalo $[0.5, 1]$, de acordo com a Equação 3.11

$$R'_{ij}(t) = \frac{R_{ij}(t)}{2D} + 0.5 \tag{3.11}$$

onde D é a maior distância entre duas soluções do sistema.

Algoritmo 8 Operação de Movimento Dependente do *DGSA*

```
i ← o índice da solução em questão
for k = 1 to K do
    j ← o índice da k-ésima pior solução do sistema
    for l = 1 to DML_ij do
        solução i ← alguma solução vizinha mais próxima da solução j
    end for
end for
return solução i
```

A solução do *DGSA* para o problema do caixeiro viajante foi comparada com a solução de outros algoritmos. O resultado indica que o *DGSA* pode ser usado em problemas discretos.

Nesse artigo, a constante gravitacional é chamada de coeficiente gravitacional, cujo valor inicial está no intervalo (0, 1], sem justificativas.

3.6 *Black hole GSA 1*

Doraghinejad et al. apresentaram duas versões do *GSA — Black hole GSA 1 (BH-GSA1)* e *Black hole GSA 2 (BH-GSA2)*, que fazem uso de uma operação chamada de buraco negro. De acordo com a teoria da relatividade, a força da gravidade em um buraco negro é tão grande que nem a luz pode escapar dele.

No *BH-GSA1*, as partículas mais pesadas do sistema tornam-se buracos negros (DORAGHINEJAD et al., 2012). O raio de um buraco negro (R_s) é dado pela Equação 3.12

$$R_s = M * log(t) \tag{3.12}$$

onde M é a massa do buraco negro.

As outras partículas são divididas em pesadas e leves. Quando uma partícula pesada se acha no raio de um buraco negro, a sua posição é atualizada de acordo com a Equação 3.13

$$x_i(t) = x_i(t)\left(exp\left(-\frac{t-1}{2M}\right)+1\right) \tag{3.13}$$

Quando uma partícula leve se acha no raio de um buraco negro, a sua posição é atualizada de acordo com a Equação 3.14

$$x_i(t) = x_i(t)R\left(\frac{1}{log(t-1)+1}\right) \tag{3.14}$$

onde R é a distância entre a partícula e o buraco negro.

O *BH-GSA1* foi comparado com o *SGSA* e com o *IGSA* em experimentos com sete funções de referência unimodais. Os resultados mostraram que o *BH-GSA1* é melhor que os outros dois algoritmos em problemas unimodais.

A constante gravitacional inicial usada nos experimentos não é especificada.

3.7 *Black hole GSA 2*

No *BH-GSA2*, só a partícula mais pesada do sistema torna-se um buraco negro (DORAGHINEJAD; NEZAMABADI-POUR, 2014). O buraco negro possui dois raios: R_s, dado pela Equação 3.12, e R'_s, dado pela Equação 3.15

$$R'_s = GMv^2/t \tag{3.15}$$

onde v é a velocidade do buraco negro.

As demais partículas são divididas em pesadas e leves. Quando uma partícula pesada se acha no raio R_s do buraco negro, a sua posição é atualizada de acordo com a Equação 3.16

$$x_i^d(t) = x_i^d(t) + random(x_{BH}(t-1) - x_i(t-1)) \tag{3.16}$$

onde x_{BH} é a posição do buraco negro.

Quando uma partícula leve se acha no raio R'_s de um buraco negro, a sua posição é atualizada de acordo com a Equação 3.17

$$x_i^d(t) = random(x_i^d(t-1)(r_i/R'_s)) \tag{3.17}$$

onde r_i é a distância entre a partícula i e o buraco negro.

O *BH-GSA2* foi comparado com quatro algoritmos — *RGA*, *PSO*, *SGSA* e *IGSA* —, em experimentos com várias funções de referência. Os resultados mostraram que a convergência do *BH-GSA2* é similar à do *IGSA*.

A constante gravitacional original $G_0 = 100$ é usada nos experimentos, sem justificativas.

3.8 *Niche GSA*

Yazdani et al. apresentaram uma versão do *GSA — Niche GSA (NGSA)* —, apropriada para problemas multimodais (YAZDANI et al., 2014). No *Niche GSA (NGSA)*, cada partícula do sistema tem um nicho que engloba K partículas. A partícula i faz parte do enxame da partícula j, se a partícula i é uma das K partículas mais próximas da partícula j. A massa gravitacional de cada partícula é um vetor, de acordo com a Equação 3.18

$$m_i(t) = (m_{i1}(t),\ m_{i2}(t),\ ...,\ m_{iN}(t)) \tag{3.18}$$

onde $m_{ij}(t)$ é a massa da partícula i no enxame da partícula j na iteração t, de acordo com a Equação 3.19

$$m_{ij}(t) = \begin{cases} \dfrac{q_i(t) - w_j(t)}{b_j(t) - w_j(t)} & \text{se a partícula } i \text{ faz parte do enxame da partícula } j \\ 0 & \text{do contrário} \end{cases} \tag{3.19}$$

onde $q_i(t)$ é a qualidade da solução i, $b_j(t)$ é a qualidade da melhor solução do enxame da partícula j e $w_j(t)$ é a qualidade da pior solução do enxame da partícula j.

Tal como no *GSA*, o número de partículas de cada enxame do *NGSA* também é uma variável em função da iteração, de acordo com a Equação 3.20

$$K(t) = round([K_i - (K_i - K_f)t/T]N) \tag{3.20}$$

onde K_i é o número inicial de partículas por enxame e K_f é o número final de partículas por enxame.

No *NGSA*, a posição de uma partícula só é atualizada se a solução da nova posição for melhor do que a anterior.

O *NGSA* foi comparado com algumas meta-heurísticas — *Deterministic Crowding*, *Sequential Niche* e algumas versões do *PSO* — apropriadas para problemas multimodais, em experimentos com várias funções de referência multimodais conhecidas. Os resultados mostraram que o *NGSA* é apropriado para problemas multimodais, mas depende muito dos valores atribuídos a K_i e K_f.

A constante gravitacional inicial usada nos experimentos é proporcional ao espaço de busca do problema: $G_0 = 0.1 * D_f$, onde D_f é o domínio da função. No entanto, essa equação não é justificada e não é válida para problemas de espaços irregulares, que são os problemas de verdade.

3.9 *GSA with negative mass*

Khajooei e Rashedi apresentaram uma versão do *GSA — GSA with Negative Mass (GSAN)* —, que leva em consideração a anti-gravidade (KHAJOOEI; RASHEDI, 2016).

Um dos maiores problemas do *GSA* é a convergência prematura, que é devida ao fato da força da gravidade só atrair partículas umas às outras, e nunca repelir. O *GSAN* é baseado na ideia de que a força da gravidade tem uma inversa, chamada de anti-gravidade. Assim, quando uma partícula de massa positiva encontra uma partícula de massa negativa, elas não se atraem, mas, ao contrário, se repelem.

A massa de uma partícula no *GSAN* é determinada pela Equação 3.21

$$m_i(t) = \frac{q_i(t)}{\sum_{j=1}^{N} |q_j(t)|} \tag{3.21}$$

onde $q_i(t)$ é a qualidade da solução i, de acordo com a Equação 3.22

$$q_i(t) = \frac{o_i(t) - w(t)}{b(t) - w(t)} - C(t) \tag{3.22}$$

Portanto, se a qualidade da solução i na iteração t é maior do que a constante C na iteração t, então a massa da partícula i é positiva, e vice-versa. A constante C é determinada pela Equação 3.23.

$$C(t) = C_0 e^{-\beta t/T} \tag{3.23}$$

O *GSAN* foi comparado com o *SGSA* em algumas funções de referência conhecidas, mas seus resultados não se mostraram significativamente melhores que os do *SGSA*.

A constante gravitacional original $G_0 = 100$ foi usada nos experimentos, sem justificativas.

3.10 *Fitness Varying Gravitational Constant GSA*

Bansal et al. apresentaram o *Fitness Varying Gravitational Constant GSA (FVGGSA)*, modificando a constante gravitacional (BANSAL et al., 2018). A execução desse algoritmo é dividida em várias fases e em cada fase a constante gravitacional tem uma escala. A constante gravitacional em determinada fase é dada pela Equação 3.24

$$G'(t) = Ze^{-\alpha t/\eta} \tag{3.24}$$

onde $G'(t)$ é a constante gravitacional na iteração t dessa fase, η é o número máximo de iterações dessa fase e Z é a constante de escala dessa fase, determinada pela Equação 3.25

$$Z(x) = 1408e^{-0.00529x} \tag{3.25}$$

onde x é a iteração média dessa fase.

Além disso, cada partícula realiza uma operação: as melhores realizam a intensificação e as piores, a diversificação. Para tanto, uma *fitness varying gravitational constant* é determinada para cada partícula, de acordo Equação 3.26

$$G_i(t) = G'(t)(1.45 - prob_i) \tag{3.26}$$

onde a variável $prob_i$ é proporcional à qualidade da solução i, de acordo com a Equação 3.27

$$prob_i = \frac{0.9 * fit(i)}{max\ fit} + 0.1 \tag{3.27}$$

onde $fit(i)$ é a qualidade da solução i, de acordo com a Equação 3.28

$$fit(i) = \begin{cases} 1 - f_i, & f_i < 0 \\ \dfrac{1}{1 + f_i}, & f_i \geq 0 \end{cases} \tag{3.28}$$

onde f_i é o valor da função objetiva da solução i.

O *FVGGSA* foi comparado com quatro algoritmos — *SGSA*, *Chaotic GSA (CGSA)*, *Biogeography-based Optimization (BBO)* e *Disruption BBO (DBBO)* — usando dois conjuntos de funções de referência. Os resultados mostraram que o *FVGGSA* é mais resistente a convergências prematuras do que o *GSA* e um pouco melhor do que os outros algoritmos.

A constante gravitacional original $G_0 = 100$ foi usada nos experimentos, sem justificativas.

3.11 Aplicações do *GSA*

Duman et al. usaram o *GSA* no problema de despacho econômico (economic dispatch) de energia em um sistema elétrico. O objetivo desse problema é programar a produção de energia de todas as unidades de um sistema, de forma tal que a demanda de combustível seja minimizada. Os experimentos mostraram que o *GSA* pode ser usado em problemas de despacho econômico, mas ficou sensível aos valores dos parâmetros. A constante gravitacional original $G_0 = 100$ foi usada nos experimentos, sem justificativas (DUMAN et al., 2010).

Rashedi et al. usaram o *GSA* na modelagem de filtros digitais. Filtros digitais são usados em várias áreas, tais como processamento de sinais, processamento de imagens, sistemas de comunicação, dentre outras. A função objetiva a ser minimizada é o erro quadrático médio da estimativa do filtro digital. O *GSA* foi comparado ao *GA* e ao *PSO* em uma série de experimentos com um filtro de resposta infinita ao impulso e um filtro não-linear racional. Os resultados indicaram que a convergência do *GSA* é comparável à das meta-heurísticas mais usadas. A constante gravitacional original $G_0 = 100$ foi usada nos experimentos (RASHEDI et al., 2011).

Li e Zhou usaram o *GSA* na configuração de um sistema de controle de turbinas hidráulicas (SCTH). O sistema é responsável pela segurança, pela estabilidade e pela eficiência de uma usina hidrelétrica. A modelagem do SCTH é dividida em quatro partes: regulador, sistema de comportas, turbina hidráulica e gerador. Cada parte tem suas próprias equações e restrições, o que torna o SCTH um sistema complexo. A equação da velocidade de uma partícula foi modificada no *GSA*, de forma a considerar além das soluções da iteração, também as melhores soluções do sistema, tal como o faz o *PSO*. O *GSA* modificado produziu boas configurações para o SCTH. As constantes gravitacionais iniciais usadas nos experimentos não foram justificadas (LI; ZHOU, 2011).

Duman et al. usaram o *GSA* no problema de cálculo do fluxo de potência ótimo (FPO) de um sistema elétrico. O FPO garante a segurança e a eficiência de operação de um sistema elétrico de potência. O cálculo do FPO envolve a otimização de funções objetivas, como, por exemplo, a minimização do custo de combustível, a minimização do custo quadrático por peça. O *GSA* foi avaliado sobre funções de referência, típicas de uma modelagem do problema. Os resultados do *GSA* foram superiores aos de outros algoritmos descritos na literatura: *Biogeography-based optimization (BBO)*, *PSO* e outros. A constante gravitacional original $G_0 = 100$ foi usada nos experimentos (DUMAN et al., 2012).

Mirjalili et al. usaram uma versão do *GSA* combinada com o *PSO* — *PSOGSA* — no treinamento de redes neurais artificiais (*Artificial Neural Networks (ANNs)*). ANNs são usadas em várias áreas, tais como em sistemas de automação, reconhecimento de padrões, reconhecimento de sequências, jogos eletrônicos, dentre outras. Técnicas clássicas são usadas no treinamento das ANNs, mas vários estudos mostram que meta-heurísticas podem também ser usadas no treinamento. O *PSOGSA* foi comparado com o *PSO* e com o *GSA* no treinamento de ANNs, apresentando melhores resultados. A constante gravitacional inicial usada nos experimentos foi $G_0 = 1$, sem justificativas (MIRJALILI et al., 2012).

Marzbanda et al. usaram o *GSA* na configuração de um sistema de geração distribuída de energia (SGDE) de uma micro-rede (MR). A modelagem de um SGDE é dividida em quatro partes — turbina de vento, células fotovoltaicas, microturbina e sistema de armazenamento de energia — e envolve restrições relativas ao equilíbrio de energia, a limitações das unidades de energia renovável e não-renovável, ao armazenamento da energia, dentre outras. Os autores fizeram uso de uma versão modificada do *GSA* — *Multi-period GSA (MGSA)* — que leva em consideração as características próprias desse problema. O *MGSA* foi testado na micro-rede do Instituto de Pesquisa de Energia da Catalunha, produzindo melhores configurações do que as produzidas pelo sistema anteriormente utilizado (MARZBANDA et al., 2014).

Chen et al. usaram uma versão modificada do *GSA* — *MGSA-NSGA-III* —, baseada no *Non-dominated sorting GA-III (NSGA-III)*, em um problema de otimização da geração

de uma usina de energia elétrica. Nesse problema, a produção de energia da usina deve ser distribuída entre as unidades geradoras hidráulicas, térmicas e eólicas. O problema envolve tanto a maximização da eficiência de operação da usina quanto a minimização da poluição produzida. Uma usina de energia com quatro unidades hidráulicas, três térmicas e duas eólicas foi modelada. O *MGSA-NSGA-III* foi comparado ao *MOGSA* e ao *NSGA-III*. Os resultados mostraram que o *MGSA-NSGA-III* pode ser usado em aplicações reais na otimização da operação de usinas geradoras de energia. A constante gravitacional original $G_0 = 100$ foi usada nos experimentos (CHEN et al., 2017).

Li et al. usaram uma versão modificada do *GSA* — *CGGSA* —, baseada na mutação de Cauchy e na mutação gaussiana do *GA*, na configuração de um controlador PID (PID controller) de uma usina hidrelétrica reversível. Esse controlador é responsável pela qualidade, confiabilidade, segurança, estabilidade e eficiência de uma usina hidrelétrica. O *CGGSA* foi comparado com o *GSA* e o *PSO* em experimentos com treze funções de referência conhecidas e com algumas estratégias clássicas de configuração de controladores PID de ordem fracionária. O *CGGSA* produziu resultados melhores do que as outras estratégias de controle. Foi realizada uma série de experimentos para determinar a constante gravitacional inicial mais apropriada, variando no intervalo $[0,\ 125]$. Concluiu-se que a constante original $G_0 = 100$ era suficientemente boa (LI et al., 2017).

Packiasudha et al. usaram uma versão do *GSA* — *Cumulative GSA (CGSA)* —, que leva em consideração as massas ativa e passiva das partículas, no posicionamento de um dispositivo de transmissão de corrente alternada flexível (*flexible alternating current transmission (FACT) device*) em uma zona industrial. Quando esse dispositivo é posto na posição correta, as perdas de energia reativa da rede elétrica podem ser controladas, o que pode aumentar o fluxo de energia ativa. Os experimentos visaram determinar a melhor posição do FACT na linha de alta voltagem do sul da Índia. O *CGSA* foi comparado com outras técnicas próprias para essa aplicação, apresentando melhores resultados. A constante gravitacional inicial usada nos experimentos não foi especificada (PACKIASUDHA et al., 2017).

Beigvand et al. usaram uma versão do *PSO* combinada com o *GSA* — *time varying acceleration coefficients-gravitational search algorithm-particle swarm optimization (TVAC-GSA-PSO)* — no problema do despacho econômico de energia combinada com calor (DEECC), que pode aumentar a eficiência de um sistema de energia de 50 a 60% para 90%. O *TVAC-GSA-PSO* foi comparado com algumas meta-heurísticas em experimentos com cinco funções objetivas e com duas modelagens distintas do DEECC. Os resultados mostraram que o *TVAC-GSA-PSO* apresenta soluções eficazes e confiáveis para o problema do DEECC. As constantes gravitacionais iniciais usadas nos experimentos foram determinadas experimentalmente (BEIGVAND et al., 2017).

Hashemi et al. usaram o *MO-GSA* na otimização de um sub-sistema de geração de energia eólica de larga escala (SGEELE) dentro de um sistema de energia de múltiplas

máquinas (SEMM). O objetivo da otimização era reduzir o impacto do cisalhamento do vento e da sombra da torre, que provocam oscilações na potência ativa da rede. O sistema proposto foi dividido em duas fases. Na primeira, um conjunto de soluções era produzido pelo *MO-GSA*. Na segunda, uma solução era selecionada por um *fuzzy optimal assessment (FOA)*. Esse sistema foi testado na modelagem citada e os resultados mostraram uma melhora significativa na estabilidade da rede. A constante gravitacional inicial usada nos experimentos $G_0 = 300$ não foi justificada (HASHEMI et al., 2017).

Moeini et al. usaram o *GSA* no Problema da Operação de Reservatórios. A otimização da operação é um problema complexo, multi-objetivo e com muitas restrições, sendo de difícil solução mesmo para algoritmos modernos. Duas operações de um reservatório foram modeladas — operação simples e operação de energia hidráulica. Três versões do *GSA* — *unconstrained GSA (UGSA)* (que é o próprio *GSA*), *partially constrained GSA (PCGSA)* e *fully constrained GSA (FCGSA)* — foram usadas. Os três algoritmos foram testados nas duas modelagens citadas. Os resultados mostraram que o *PCGSA* e o *FCGSA* produziram melhores soluções do que o *UGSA*. A constante gravitacional inicial usada nos experimentos $G_0 = 250$ foi determinada experimentalmente (MOEINI et al., 2017).

3.12 Considerações Finais

O *GSA* tem vários problemas conhecidos na literatura, como a suscetibilidade à convergência prematura e a sensibilidade à constante gravitacional inicial, por exemplo (SABRI et al., 2013). A maioria das aplicações desse algoritmo usa uma constante gravitacional inicial determinada por tentativas e correções, porque não há uma fórmula para determinar a constante mais apropriada para cada aplicação. A única fórmula conhecida não é válida para problemas de espaços irregulares, que são os problemas reais (YAZDANI et al., 2014). Além disso, nenhuma aplicação da literatura considera essa fórmula. Neste trabalho, é proposta uma heurística para determinar a constante gravitacional inicial do *GSA*, conforme o espaço de busca do problema.

4 Desenvolvimento

Conforme dito no Capítulo 1, a convergência do *GSA* depende da constante gravitacional inicial, que é determinada empiricamente. Neste capítulo, é proposta uma heurística para determinar essa constante: a *NGC*. Na seção 4.1, a *NGC* é explicada pela Teoria de Brans-Dicke. Na seção 4.2, a observação de que a constante gravitacional inicial deve ser proporcional ao espaço de busca é confirmada, justificando a *NGC*. Na seção 4.3, a constante β dessa heurística é determinada. Na seção 4.4, é feita uma revisão dessa heurística.

4.1 Da Teoria de Brans-Dicke

As teorias da dinâmica são divididas em duas linhas: a newtoniana e a machiana. Isaac Newton acreditava que o espaço é uma estrutura física independente da matéria, imutável e imóvel; definindo assim o espaço relativo, que nós determinamos pelas posições relativas dos corpos; e o espaço absoluto, que é o espaço em si (NEWTON, 1846, p. 77). Em contrapartida, Ernst Mach acreditava que as características geométricas do espaço são determinadas pela matéria. Na Teoria Machiana, o espaço relativo é o único que nós podemos considerar, pois o espaço absoluto não passa de uma conjectura.

> Ninguém é competente para falar das características do espaço absoluto e do movimento absoluto, que são puras coisas do pensamento; puras construções mentais, que não podem ser produzidas na experiência. Todos os princípios da mecânica são, como nós temos mostrado em detalhes, conhecimento experimental sobre as posições relativas e os movimentos relativos dos corpos. [...] Ninguém pode estender esses princípios além dos limites da experiência. (MACH, 1919, p. 229)

Além da geometria do espaço, os defensores da teoria machiana também acreditam que as leis da dinâmica — como a força inercial e a força gravitacional, por exemplo — são determinadas pela disposição da matéria do universo. A maior limitação dessa teoria é que ela não dá uma relação quantitativa entre essas forças e a massa (D'INVERNO, 1992, p. 124). Várias soluções foram propostas para essa questão, entre as quais se destaca a Teoria de Brans-Dicke. Essa teoria diz que a força gravitacional é diretamente proporcional ao raio do universo visível e inversamente proporcional à massa (BRANS; DICKE, 1961), de acordo com a Equação 4.1

$$G \sim \frac{Rc^2}{M} \tag{4.1}$$

onde G é a constante gravitacional, R e M são o raio e a massa do universo visível, respectivamente, e c é a velocidade da luz no vácuo.

Usando um sistema de unidades geométricas, a Equação 4.1 é simplificada para a Equação 4.2

$$G \sim \frac{R}{M} \tag{4.2}$$

onde G é a constante gravitacional e R e M são o raio e a massa do universo visível, respectivamente.

É proposta uma heurística para determinar a constante gravitacional inicial do *GSA* baseada nessa teoria. Considerando que a massa total do *GSA* é normalizada, a Constante Gravitacional Normalizada — *Normalized Gravitational Constant (NGC)* — é proporcional à média artimética dos tamanhos das dimensões do espaço de busca do problema, de acordo com a Equação 4.3

$$G_0 = \sum_{i=1}^{N} (L_i^+ - L_i^-) * \beta / N \tag{4.3}$$

onde G_0 é a *NGC*, L_i^+ e L_i^- são os limites máximo e mínimo da dimensão i do espaço de busca, respectivamente, β é a constante de proporcionalidade dessa relação, que é determinada na seção 4.3, e N é o número de variáveis do problema.

Graças à Teoria de Brans-Dicke, o Algoritmo de Busca Gravitacional com a Constante Gravitacional Normalizada — *Gravitational Search Algorithm with Normalized Gravitational Constant (GSA-NGC)* — é mais robusto e adaptável a espaços de busca irregulares do que o algoritmo original.

4.2 Da constante gravitacional inicial do *GSA* e do espaço de busca do problema

Alega-se que a constante gravitacional inicial deve ser proporcional ao espaço de busca do problema. Essa alegação é justificada por duas observações — a primeira, que a velocidade ótima na primeira iteração é proporcional ao espaço de busca, tratada na subseção 4.2.1; e a segunda, que a aceleração é proporcional à constante gravitacional e limitada por ela, tratada na subseção 4.2.2 — e confirmada experimentalmente na subseção 4.2.3.

4.2.1 Da movimentação de uma partícula do *GSA* e do espaço de busca do problema

Conforme dito no Capítulo 2, a velocidade das partículas do *GSA* deve ser grande nas primeiras iterações e pequena nas últimas. Na primeira iteração, ela não pode ser muito grande, para que as partículas não fiquem pulando o ótimo global; nem muito pequena, para que a convergência não seja muito lenta. A velocidade ótima depende da função

objetiva e, portanto, não pode ser determinada a priori. No entanto, pode ser feita uma aproximação conforme o espaço de busca, considerando duas condições extremas:

1. Se a velocidade das partículas, na primeira iteração, for maior do que o espaço de busca; então vão acontecer muitas fugas, conforme a Figura 10, independentemente da função objetiva. Essa condição vai continuar durante muitas iterações, até a constante gravitacional alcançar um valor apropriado.

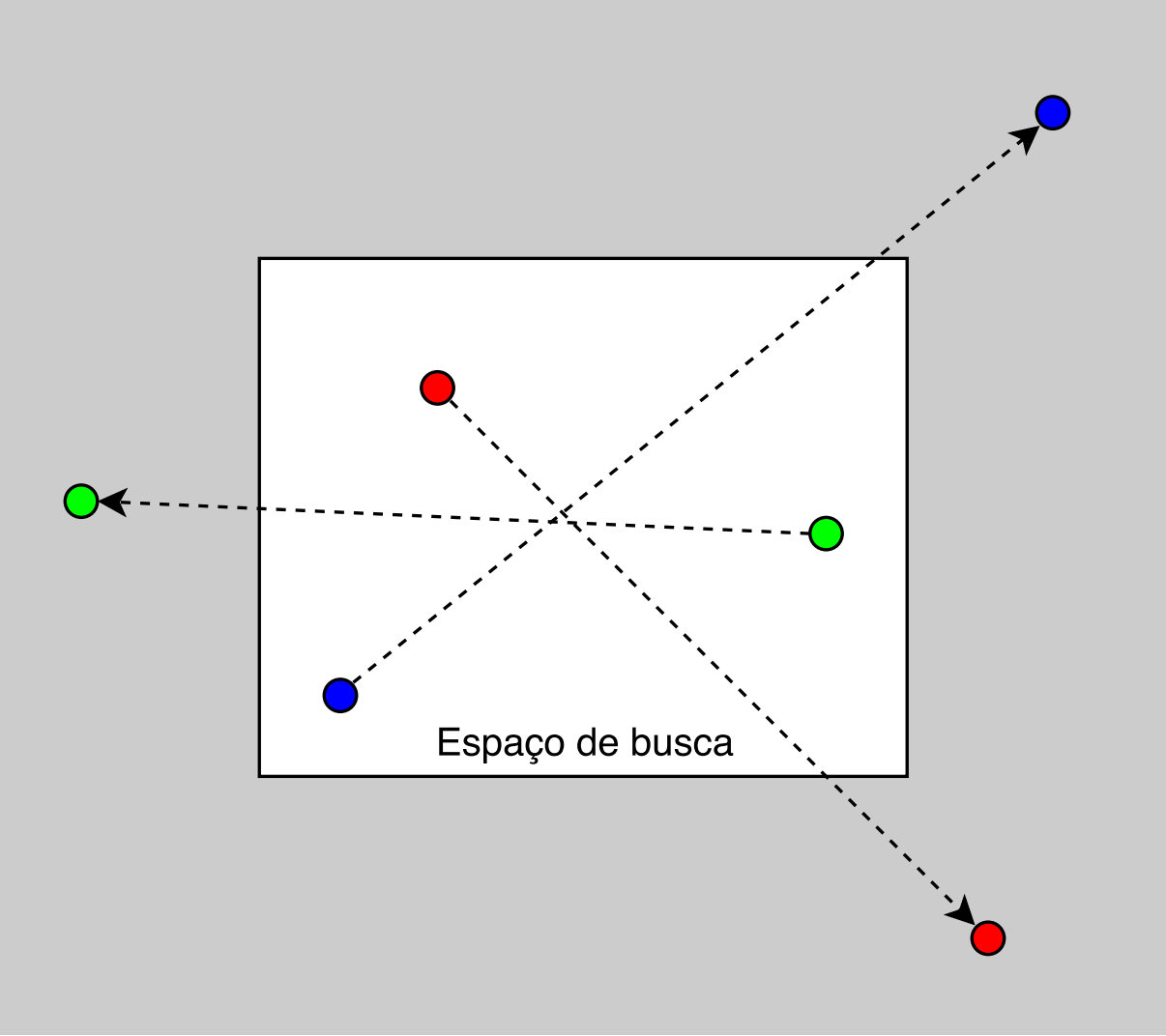

Figura 10 – Muitas partículas em fuga

2. Em contrapartida, se ela for muito menor do que o espaço de busca, então a exploração vai ser praticamente nula. E essa condição só vai se agravar, porque a constante gravitacional só vai diminuir.

Ou seja, uma velocidade maior do que o espaço de busca vai provocar muitas fugas e não vai ajudar na exploração, independentemente da função objetiva, enquanto uma

muito menor pode provocar a convergência prematura ou até a estagnação. Embora a velocidade ótima não possa ser determinada a priori, contudo, é evidente que ela não pode ser nem maior nem muito menor do que o espaço de busca, sendo assim proporcional a ele.

4.2.2 Da aceleração de uma partícula do *GSA* e da constante gravitacional

Sabe-se que a aceleração de uma partícula do *GSA* a é proporcional à constante gravitacional G, ou seja: $a \propto G$. Além disso, demonstra-se que ela é limitada pela constante gravitacional, ou seja: $a \leq G$, independentemente da função objetiva. Essa demonstração é feita em duas partes:

1. Considerando que a aceleração de uma partícula é igual à força total sobre ela (Equação 2.24), para que a aceleração da partícula i seja a máxima, é necessário que todas as partículas estejam na mesma linha, conforme a Figura 11.

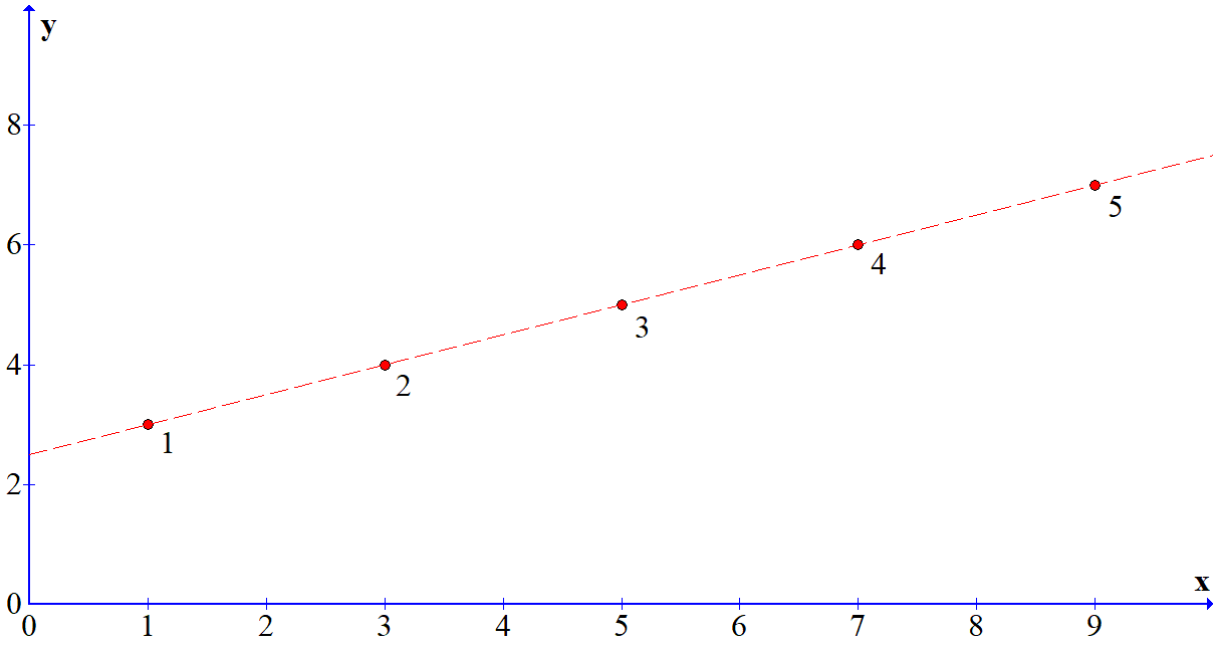

Figura 11 – Todas as partículas na mesma linha (condição extrema)

Essa condição não é de forma alguma comum, mas deve ser suposta para determinar a aceleração máxima de uma partícula. De fato, supondo que todas as partículas estão na mesma linha, delimitada pelas partículas i e k, a força total sobre a partícula i é de acordo com a Equação 4.4

$$F_i(t) = G(t) < x_k(t) - x_i(t) > \sum_{j=1}^{P} (u(D) \circ m_j(t)), \quad j \neq i \tag{4.4}$$

onde $F_i(t)$ é a força total sobre a partícula i na iteração t, $G(t)$ é a constante gravitacional na iteração t; $x_i(t)$ e $x_k(t)$ são as posições das partículas i, j e k na iteração t, respectivamente; $u(D)$ é um vetor de D variáveis aleatórias no intervalo $[0,\ 1]$, $m_j(t)$ é a massa da partícula j na iteração t e P é o número de partículas do algoritmo.

2. Analisando essa equação, observa-se que a força total sobre a partícula i é máxima quando a solução i é a pior — e, portanto, a massa da partícula i é igual a zero e a somatória das massas de todas as outras partículas é igual a 1 — e todas as variáveis aleatórias de $u(D)$ valem 1.

   Supondo que todas essas condições sejam atendidas, portanto, a Equação 4.4 é simplificada para a Equação 4.5

$$F_i(t) = G(t) < x_k(t) - x_i(t) > \tag{4.5}$$

   onde $F_i(t)$ é a força total sobre a partícula i na iteração t, $G(t)$ é a constante gravitacional na iteração t, e $x_i(t)$ e $x_k(t)$ são as posições das partículas i e k na iteração t, respectivamente.

   Analisando essa equação, observa-se que a magnitude da aceleração máxima de uma partícula — ou seja, a magnitude máxima da aceleração de uma partícula — é igual à constante gravitacional.

Ou seja, a velocidade ótima na primeira iteração é proporcional ao espaço de busca do problema, independentemente da função objetiva, e a aceleração é proporcional à constante gravitacional e limitada por ela, sugerindo que a constante gravitacional inicial do *GSA* deve ser proporcional ao espaço de busca do problema.

4.2.3 Experimentos variando a constante gravitacional inicial do *GSA*

Foram realizados experimentos para determinar a relação entre a constante gravitacional inicial do *GSA* e o espaço de busca do problema. A implementação do *GSA* em *Octave* é mostrada no Apêndice A.

Foram usadas treze funções de referência conhecidas na literatura, sete unimodais: *Sphere Function* (1), *Schwefel Function 1* (2), *Schwefel Function 2* (3), *Schwefel Function 3* (4), *Rosenbrock Function* (5), *Step Function* (6) e *Quartic Function* (7); e seis multimodais: *Schwefel Function 4* (8), *Rastrigin Function* (9), *Ackley Function* (10), *Griewank Function* (11), *Penalized Function 1* (12) e *Penalized Function 2* (13). As funções unimodais são usadas para avaliar a intensificação do algoritmo e as multimodais, para avaliar o equilíbrio entre a intensificação e a diversificação. Além disso, cada função tem uma paisagem (*fitness landscape*) distinta, para avaliar a versatilidade do algoritmo.

Essas funções são mostradas na Tabela 1; os ótimos, as soluções e os limites mínimos e máximos das dimensões dos espaços de busca são mostrados na Tabela 2; os gráficos dessas funções em duas dimensões são mostrados no Apêndice B e a implementação dessas funções em *Octave* é mostrada no Apêndice C.

Índice	Função
1	$f_1(X) = \sum_{i=1}^{N} X_i^2$
2	$f_2(X) = \sum_{i=1}^{N} \lvert X_i \rvert + \prod_{i=1}^{N} \lvert X_i \rvert$
3	$f_3(X) = \sum_{i=1}^{N} (\sum_{j=1}^{i} x_j)^2$
4	$f_4(X) = max(\lvert X_1 \rvert, \lvert X_2 \rvert, ..., \lvert X_N \rvert)$
5	$f_5(X) = \sum_{i=2}^{N} [100(X_i - X_{i-1}^2)^2 + (X_{i-1} - 1)^2]$
6	$f_6(X) = \sum_{i=1}^{N} \lfloor X_i + 0.5 \rfloor^2$
7	$f_7(X) = \sum_{i=1}^{N} (iX_i^4) + u(1)$
8	$f_8(X) = -\sum_{i=1}^{N} (X_i sin(\sqrt{\lvert X_i \rvert}))$
9	$f_9(X) = \sum_{i=1}^{N} (X_i^2 - 10cos(2X_i\pi) + 10)$
10	$f_{10}(X) = -20exp(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} X_i^2}) - exp(\frac{1}{N}\sum_{i=1}^{N} cos(2X_i\pi)) + e + 20$
11	$f_{11}(X) = \frac{1}{4000}\sum_{i=1}^{N} X_i^2 - \prod_{i=1}^{N} cos(\frac{X_i}{\sqrt{i}}) + 1$
12	$f_{12}(X) = \frac{\pi}{N}\{10sin^2(y_1\pi) + \sum_{i=2}^{N}[(y_{i-1} - 1)^2(10sin^2(y_i\pi) + 1)] + (y_N - 1)^2\} + \sum_{i=1}^{N} u(X_i, 10, 100, 4)$
13	$f_{13}(X) = 0.1\{sin^2(3X_1\pi) + \sum_{i=2}^{N}[(X_{i-1} - 1)^2(sin^2(3X_i\pi) + 1)] + (X_N - 1)^2(sin^2(2X_N\pi) + 1)\} + \sum_{i=1}^{N} u(X_i, 5, 100, 4)$

Tabela 1 – Funções de referência

Foram realizadas cinco séries de experimentos, variando a constante gravitacional nos valores 1, 10, 100, 1000 e 10000. Foram usados os seguintes argumentos: número de

Função	Ótimo	Solução	Limite Mínimo	Limite Máximo
1	0	(0, 0, ..., 0)	-100	$+100$
2	0	(0, 0, ..., 0)	-10	$+10$
3	0	(0, 0, ..., 0)	-100	$+100$
4	0	(0, 0, ..., 0)	-100	$+100$
5	0	(1, 1, ..., 1)	-30	$+30$
6	0	(0, 0, ..., 0)	-100	$+100$
7	0	(0, 0, ..., 0)	-1.28	$+1.28$
8	-12569.5	(420.9687, 420.9687, ..., 420.9687)	-500	$+500$
9	0	(0, 0, ..., 0)	-5.12	$+5.12$
10	0	(0, 0, ..., 0)	-32	$+32$
11	0	(0, 0, ..., 0)	-600	$+600$
12	0	(1, 1, ..., 1)	-50	$+50$
13	0	(1, 1, ..., 1)	-50	$+50$

Tabela 2 – Ótimos, soluções e limites mínimos e máximos do espaço de busca das funções de referência

particulas $P = 50$, número de iterações $T = 1000$, número de dimensões $D = 30$ e $\alpha = 20$. Foram realizadas 30 execuções do *GSA* com cada constante.

A média aritmética da convergência da função 8 e as médias geométricas das convergências das outras funções são mostradas nas figuras 12 e 13. A média geométrica é uma medida mais significativa do que a mediana e proporciona uma visualização da convergência melhor do que a média aritmética. Figuras mais detalhadas são mostradas no Apêndice D. Só as 500 primeiras iterações da otimização das funções 6 e 8 são mostradas nos gráficos, porque as soluções dessas funções só convergiram nas primeiras iterações. A média aritmética da solução da função 8 e as médias geométricas das soluções das outras funções são mostradas na Tabela 3.

Função	$G_0 = 1$	$G_0 = 10$	$G_0 = 100$	$G_0 = 1000$	$G_0 = 10000$
1	$5.1141e+4$	$1.3413e+3$	$\mathbf{2.096e-17}$	$1.8345e-15$	$2.019e-13$
2	$1.6755e+1$	$\mathbf{2.2921e-9}$	$2.3753e-8$	$2.3486e-7$	$2.2617e-6$
3	$9.6171e+4$	$3.0891e+3$	$\mathbf{2.1442e+2}$	$6.3952e+2$	$4.6823e+2$
4	$7.6361e+1$	$1.3211e+1$	$\mathbf{3.2816e-9}$	$3.2566e-8$	$3.1258e-7$
5	$3.7146e+7$	$6.8019e+1$	$2.6058e+1$	$\mathbf{2.5419e+1}$	$2.5763e+1$
6	$5.3321e+4$	$1.4400e+3$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
7	$5.1894e-2$	$\mathbf{1.9075e-2}$	$1.9978e-2$	$2.4308e-2$	$3.5087e-2$
8	$-2.6360e+3$	$-2.7237e+3$	$-2.7760e+3$	$-2.4327e+3$	$\mathbf{-3.3541e+3}$
9	$1.6636e+1$	$\mathbf{7.6264}$	$1.5590e+1$	$1.6710e+1$	$1.9317e+1$
10	$1.8732e+1$	5.6425	$\mathbf{3.5378e-9}$	$3.4841e-8$	$3.5304e-7$
11	$5.6916e+2$	$3.7555e+2$	3.5594	$\mathbf{6.5512e-16}$	$6.3071e-15$
12	$1.0798e+8$	2.2418	$\mathbf{3.1655e-17}$	$1.7832e-15$	$1.0791e-14$
13	$2.6479e+8$	$3.2875e+1$	$\mathbf{9.3234e-17}$	$2.1266e-16$	$5.073e-14$

Tabela 3 – As funções de referência variando a constante gravitacional inicial

A constante $G_0 = 100$ foi a melhor em sete funções: 1, 3, 4, 6, 10, 12 e 13; convergindo nas ordens de magnitude -17, $+2$, -9, 0, -9, -17 e -17, respectivamente. A constante $G_0 = 10$ foi a melhor em três funções: 2, 7 e 9; convergindo nas ordens de magnitude -9, -2 e 0, respectivamente. A constante $G_0 = 1000$ também foi a melhor em três funções: 5, 6 e 11; convergindo nas ordens de magnitude $+1$, 0 e -16, respectivamente. A constante $G_0 = 10000$ só foi a melhor na função 8, convergindo na ordem de magnitude $+3$. E a constante $G_0 = 1$ não foi a melhor em nenhuma função.

Nas funções 1, 2, 3, 4, 5, 6, 7, 9, 10, 12 e 13, a convergência das constantes $G_0 = 1000$ e 10000 é pequena nas primeiras iterações e grande nas últimas. Essa observação é explicada pela variação da constante gravitacional: A convergência é pequena nas primeiras iterações, porque a constante gravitacional é muito grande e muitas partículas pulam os ótimos ou saem em fuga, mas ela aumenta quando a constante alcança um valor apropriado. No entanto, na função 11, essas constantes são melhores do que as outras, porque os espaços de busca dessa função é maior.

Em contrapartida, nas funções 1, 3, 4, 5, 6, 10, 11, 12 e 13, a convergência das constantes $G_0 = 1$ e 10 é pequena durante toda a execução do algoritmo. Essa observação também é explicada pela variação da constante gravitacional: A convergência é pequena durante toda a execução do algoritmo, porque a constante gravitacional é muito pequena nas primeiras iterações e ainda mais nas últimas. No entanto, nas funções 2, 7 e 9, essa constante é melhor do que as outras, porque os espaços de busca dessas funções são menores.

Na função 6, as constantes $G_0 = 100$, 1000 e 10000 convergiram em uma solução ótima exata em aproximadamente 100, 200 e 300 iterações, respectivamente, enquanto as constantes $G_0 = 1$ e 10 não convergiram. Em contrapartida, nas funções 3, 5, 8 e 9, nenhuma constante convergiu. A função 8 é especialmente complicada para o *GSA*, porque o ótimo global dessa função está nas extremidades do espaço de busca, que o *GSA* não explora bem.

Os resultados confirmaram que, se a constante gravitacional inicial for muito grande, então a velocidade das partículas também será muito grande durante as primeiras iterações da execução, e a convergência será comprometida. Em contrapartida, se ela for muito pequena, então a velocidade também será muito pequena durante toda a execução, e a convergência será ainda mais comprometida. As melhores soluções foram obtidas pelas constantes mais próximas dos tamanho das dimensões de cada espaço de busca, de uma forma geral. Os resultados confirmaram que a constante gravitacional inicial do *GSA* deve ser proporcional ao espaço de busca do problema, justificando a *NGC*.

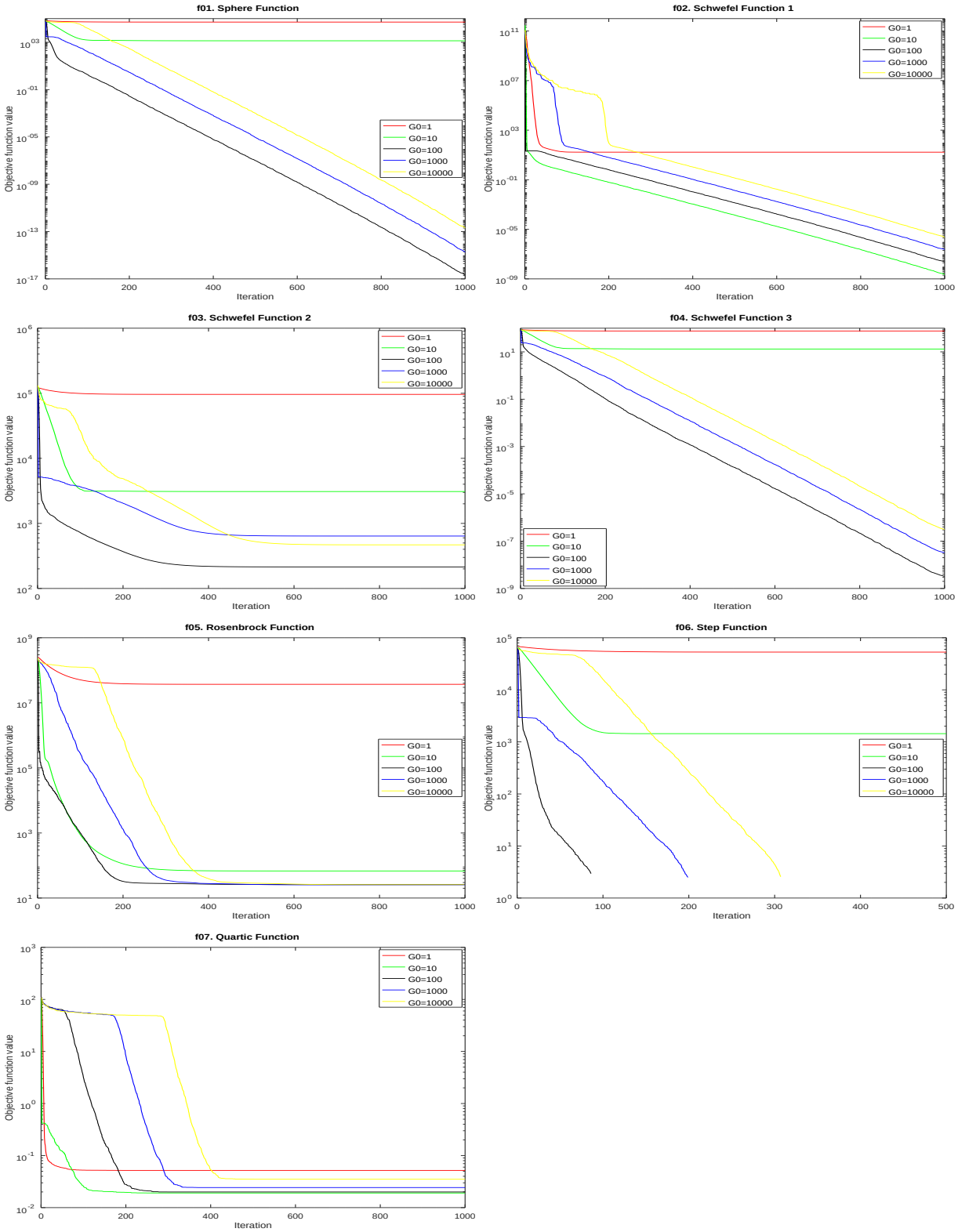

Figura 12 – As funções unimodais variando a constante gravitacional inicial

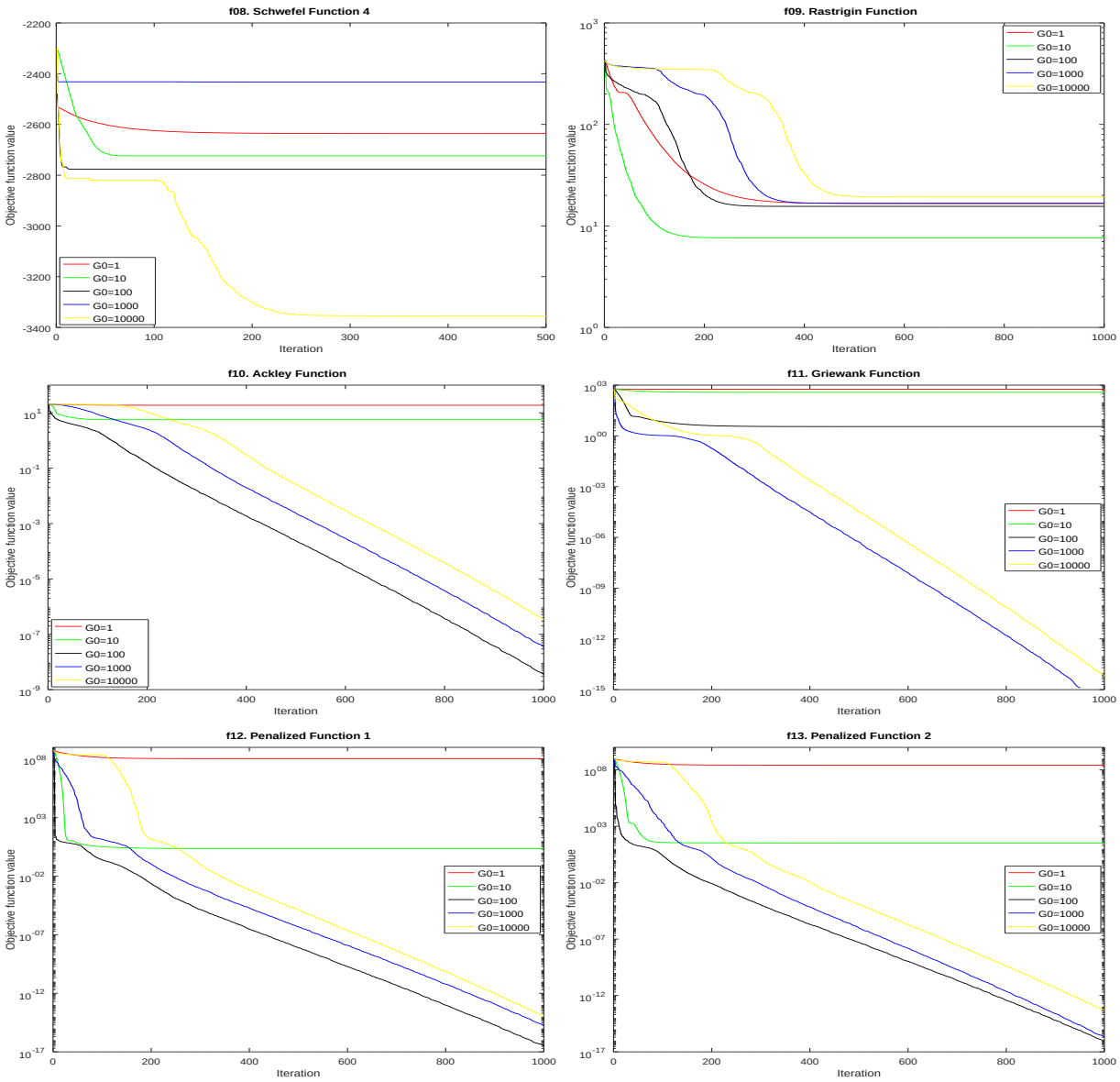

Figura 13 – As funções multimodais variando a constante gravitacional inicial

4.3 Da constante de proporcionalidade da *NGC*

Sete séries de experimentos foram realizadas para determinar a constante β mais apropriada da *NGC* (Equação 4.3), variando nos valores 0.125, 0.25, 0.5, 1, 2, 4 e 8. Foram usadas as mesmas funções de referência mostradas na Tabela 1. Foram usados os seguintes argumentos: número de particulas $P = 50$, número de iterações $T = 1000$, número de dimensões $D = 30$ e $\alpha = 20$. As constantes gravitacionais normalizadas de cada experimento são mostradas na Tabela 4.

Função	$\beta = 0.125$	$\beta = 0.25$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$	$\beta = 4$	$\beta = 8$
1	25	50	100	200	400	800	1600
2	2.5	5	10	20	40	80	160
3	25	50	100	200	400	800	1600
4	25	50	100	200	400	800	1600
5	7.5	15	30	60	120	240	480
6	25	50	100	200	400	800	1600
7	0.32	0.64	1.28	2.56	5.12	10.24	20.48
8	125	250	500	1000	2000	4000	8000
9	1.28	2.56	5.12	10.24	20.48	40.96	81.92
10	8	16	32	64	128	256	512
11	150	300	600	1200	2400	4800	9600
12	12.5	25	50	100	200	400	800
13	12.5	25	50	100	200	400	800

Tabela 4 – Constantes gravitacionais normalizadas variando a constante β

Foram realizadas 30 execuções do *GSA-NGC* com cada constante. A média aritmética da convergência da função 8 e as médias geométricas das convergências das outras funções são mostradas nas figuras 14 e 15. Figuras mais detalhadas são mostradas no Apêndice E. Só as 250 e 500 primeiras iterações da otimização das funções 6 e 8 são mostradas nos gráficos, respectivamente. A média aritmética da solução da função 8 e as médias geométricas das soluções das outras funções são mostradas na Tabela 5.

Função	$\beta = 0.125$	$\beta = 0.25$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$	$\beta = 4$	$\beta = 8$
1	*1.3599e−8*	**5.3389e−18**	2.107e−017	8.1714e−17	3.3475e−16	1.3119e−15	5.5299e−15
2	*7.3137e−1*	**2.1712e−9**	2.2372e−9	4.6915e−9	9.339e−9	1.9557e−8	3.5713e−8
3	4.7587e+2	2.9554e+2	2.3724e+2	**2.1971e+2**	4.0093e+2	*6.3198e+2*	6.2781e+2
4	*7.5510*	1.2665	**3.1007e−9**	6.2955e−9	1.2524e−8	2.4962e−8	5.0208e−8
5	*3.5013e+2*	3.600e+1	2.6602e+1	2.6664e+1	2.7896e+1	2.6982e+1	**2.5738e+1**
6	*2.7773e+2*	**0**	**0**	**0**	**0**	**0**	**0**
7	*8.8352e−2*	6.3546e−2	4.3361e−2	1.7258e−2	**1.5267e−2**	1.7875e−2	1.7551e−2
8	−2.5884e+3	−2.8147e+3	−2.7307e+3	*−2.4462e+3*	−2.6851e+3	−3.2117e+3	**−3.2680e+3**
9	**6.4424**	6.5291	6.6644	8.1304	1.1634e+1	1.2940e+1	*1.3187e+1*
10	*7.1229*	2.1282e−8	**1.1419e−9**	2.2832e−9	4.4431e−9	8.6187e−9	1.8120e−8
11	*2.2833e−1*	2.0748e−2	1.1696e−9	4.8045e−15	**2.8984e−16**	3.9428e−15	6.2681e−15
12	*2.2884*	1.4927	5.4368e−7	1.3393e−16	**1.8198e−18**	1.3062e−15	9.5627e−17
13	*2.9307e+1*	2.6607	4.0881e−12	**2.2543e−17**	2.5413e−17	3.2931e−17	3.9909e−16

Tabela 5 – As funções de referência variando a constante β da heurística

A constante $\beta = 0.125$ foi a melhor na função 9, convergindo na ordem de magnitude zero. A constante $\beta = 0.25$ foi a melhor em duas funções: 1 e 2, convergindo nas ordens de magnitude -18 e -9, respectivamente. A constante $\beta = 0.5$ foi a melhor em duas funções: 4 e 10, convergindo na ordem de magnitude -9. A constante $\beta = 1$ foi a melhor em duas funções: 3 e 13, convergindo nas ordens de magnitude $+2$ e -17, respectivamente. A constante $\beta = 2$ foi a melhor em três funções: 7, 11 e 12; convergindo nas ordens de magnitude -2, -16 e -18, respectivamente. A constante $\beta = 4$ não foi a melhor em nenhuma função. E a constante $\beta = 8$ foi a melhor em duas funções: 5 e 8, convergindo nas ordens de magnitude $+1$ e $+3$, respectivamente.

Na função 6, as constantes $\beta = 0.25$, 0.5, 1, 2, 4 e 8 convergiram em uma solução ótima exata — as constantes $\beta = 0.25$ e 0.5, em até 100 iterações; as constantes $\beta = 1$, 2 e 4, em até 200 iterações; e a constante $\beta = 8$, em até 250 iterações — enquanto a constante $\beta = 0.125$ não convergiu. Em contrapartida, nas funções 3, 5, 8 e 9, nenhuma constante convergiu.

Os resultados não apontam para uma constante β melhor do que as outras na maioria dos problemas, mas as constantes $\beta = 0.5$, 1, 2, 4 e 8 convergiram em nove funções: 1, 2, 4, 6, 7, 10, 11, 12 e 13, obtendo soluções menores do que uma unidade; enquanto a constante $\beta = 0.125$ só convergiu em quatro funções: 1, 2, 7 e 11; e a constante $\beta = 0.25$ só convergiu em seis funções: 1, 2, 6, 7, 10 e 11. Além disso, as constantes $\beta = 1$, 2, 3 e 4 convergiram em soluções oito e quatro ordens de magnitude melhores do que a constante $\beta = 0.5$ nas funções 12 e 13, respectivamente.

Tirando as constantes $\beta = 0.125$, 0.25 e 0.5, a constante $\beta = 1$ foi a melhor em oito funções: nas funções 1, 2, 3, 4, 9, 10 e 13; convergindo nas ordens de magnitude -17, -9, $+2$, -9, 0, -9 e -17, respectivamente; e na função 6, convergindo em uma solução ótima exata em aproximadamente 125 iterações. Portanto, a constante $\beta = 1$ é a melhor nessa análise. De qualquer forma, o *GSA-NGC* não é muito sensível à constante de proporcionalidade β no intervalo de 1 a 8, com as soluções variando em até três ordens de magnitude.

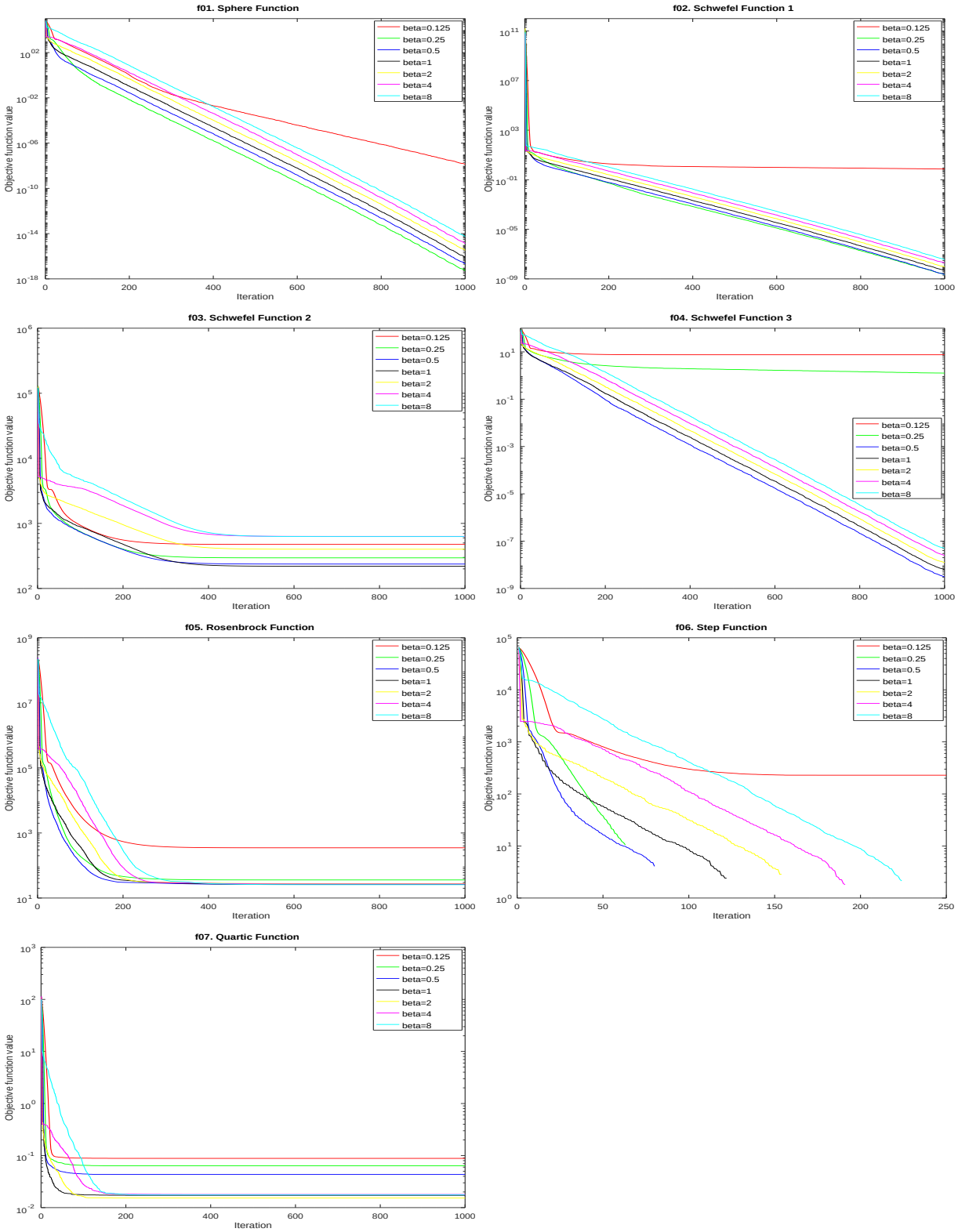

Figura 14 – As funções unimodais variando a constante β da heurística

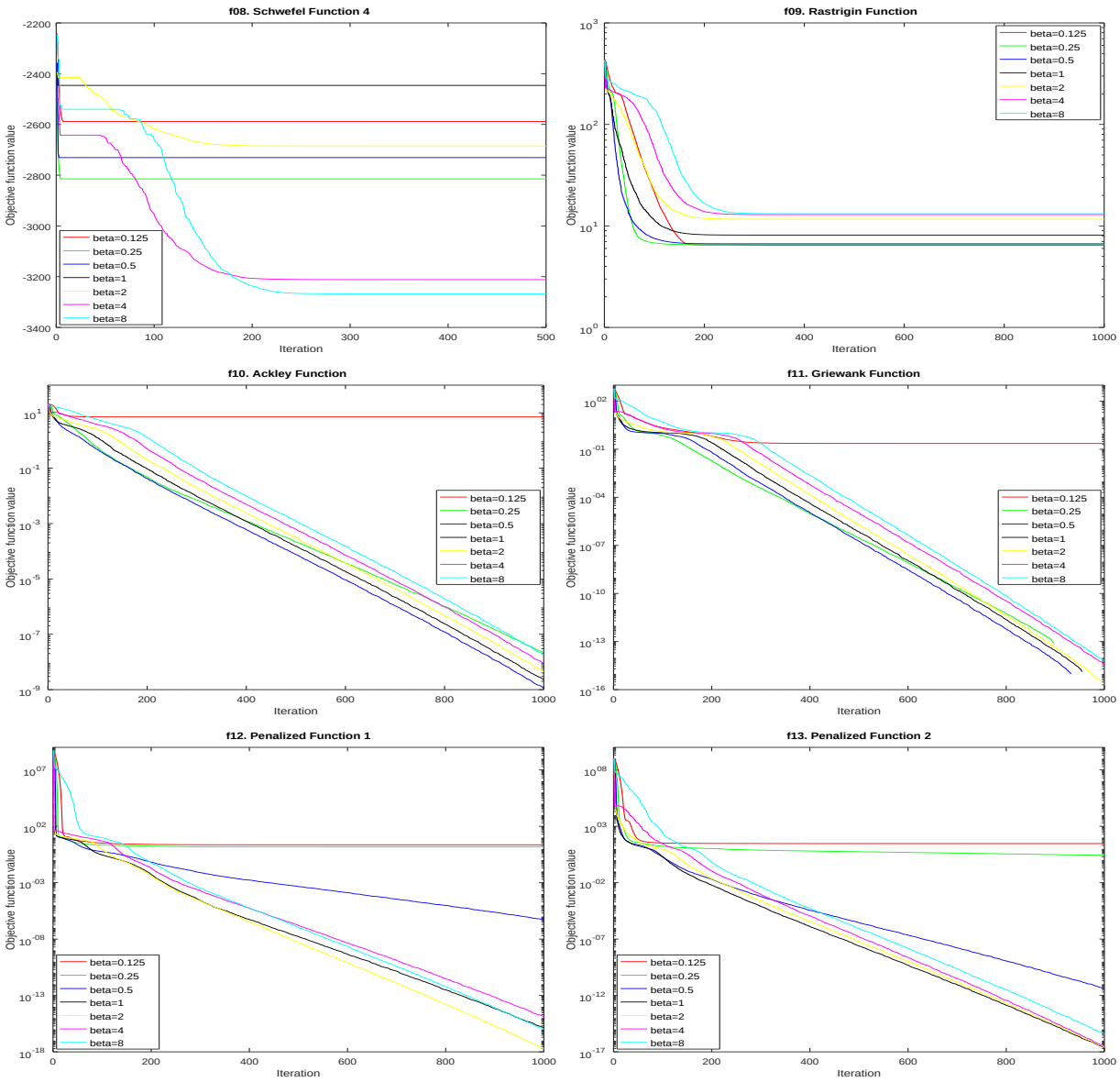

Figura 15 – As funções multimodais variando a constante β da heurística

4.4 Considerações Finais

Foi proposta uma heurística para determinar uma constante gravitacional do *GSA*, que seja proporcional ao espaço de busca do problema: a *NGC*. Essa heurística é baseada na Teoria de Brans-Dicke, que diz que a constante gravitacional é diretamente proporcional ao raio do universo visível e inversamente proporcional à massa. A observação de que a constante gravitacional inicial do *GSA* deve ser proporcional ao espaço de busca do problema foi justificada por uma análise matemática da velocidade e da aceleração das partículas, da constante gravitacional e do espaço de busca, e confirmada experimentalmente.

Importa destacar que a constante gravitacional é uma função decrescente proporcional à constante gravitacional inicial, de acordo com a Equação 2.21. A relação entre a constante gravitacional em determinada iteração e a constante gravitacional inicial é determinada pela constante α, de acordo com a Figura 16. Mudando o valor inicial, a constante gravitacional em cada iteração também é modificada, mas essas características são mantidas.

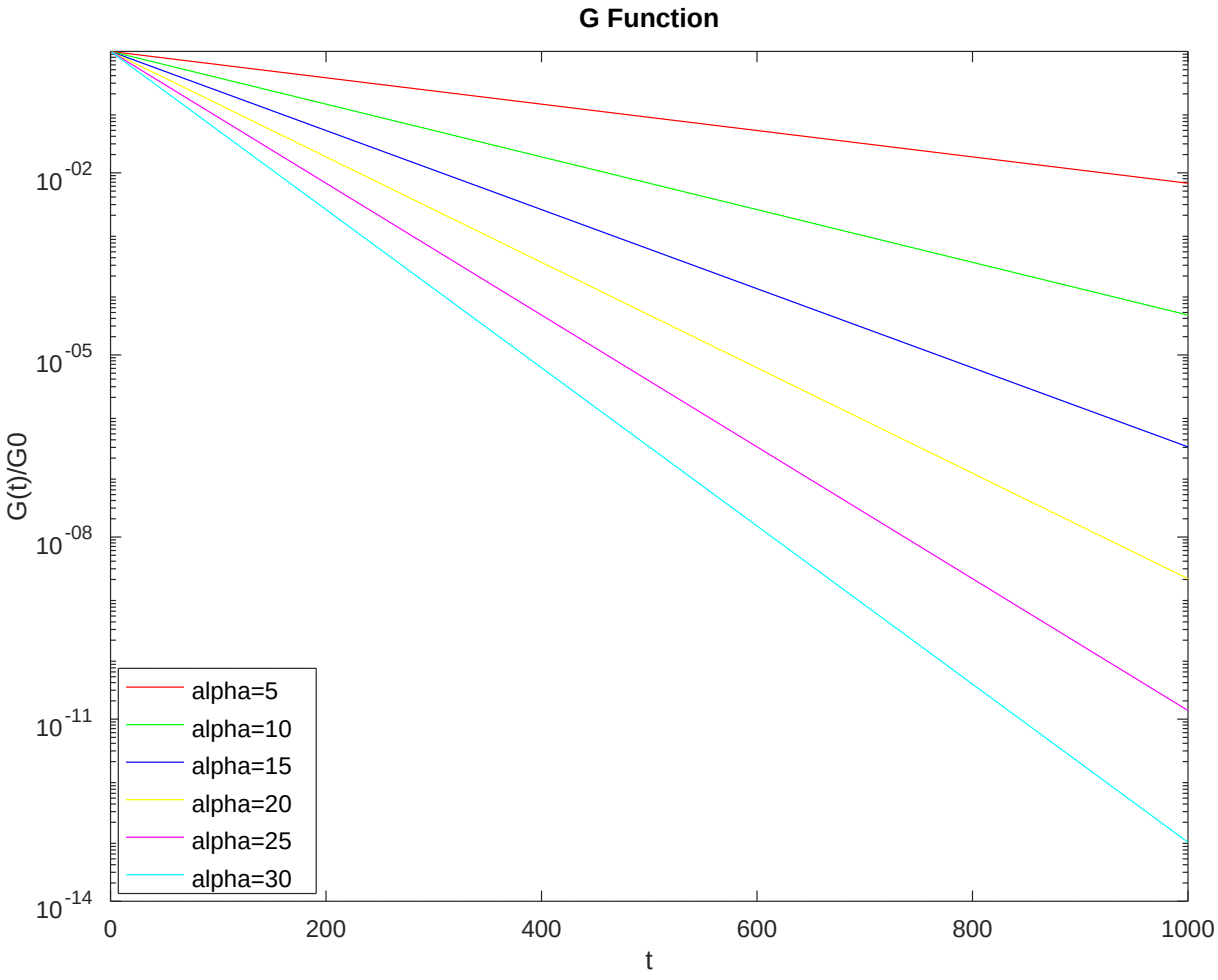

Figura 16 – A razão entre a constante gravitacional em determinada iteração $G(t)$ e a constante gravitacional inicial G_0, variando a constante α

5 Experimentos e resultados

Usando a heurística proposta, o *Gravitacional Search Algorithm with Normalized Gravitational Constant (GSA-NGC)* (Capítulo 4) com $\beta = 1$ é comparado com o *Standard GSA (SGSA)* com $G_0 = 100$, que é o valor usado por Rashedi et al. Foram usadas as mesmas funções de referência mostradas na Tabela 1 e foram realizadas três séries de experimentos: a primeira, usando um espaço de busca pequeno; a segunda, usando um espaço de busca grande; e a terceira, usando um espaço de busca irregular.

São analisadas a qualidade da solução e a velocidade da convergência. O objetivo desses experimentos é demonstrar que a *NGC* é apropriada para qualquer espaço de busca e que o *GSA* é muito sensível à constante gravitacional inicial. A análise do tempo não é necessária, porque a parte iterativa do *GSA-NGC* é igual à do *GSA*, mudando só a inicialização. As três séries de experimentos são mostradas e analisadas nas seções 5.1, 5.2 e 5.3, e uma revisão é feita na seção 5.4.

5.1 Experimentos usando um espaço de busca pequeno

Na primeira série de experimentos, as funções de referência são avaliadas em um espaço de busca relativamente pequeno, onde cada dimensão é 100 vezes menor do que a original (Tabela 2). Os limites do espaço de busca de cada função e as constantes gravitacionais normalizadas são mostradas na Tabela 6. O objetivo desses experimentos é demonstrar que a *NGC* é apropriada para problemas com espaços de busca muito pequenos.

Função	Limite Mínimo	Limite Máximo	G_0
1	-1	$+1$	2
2	-0.1	$+0.1$	0.2
3	-1	$+1$	2
4	-1	$+1$	2
5	0.7	1.3	0.6
6	-1	$+1$	2
7	$-1.28e-2$	$+1.28e-2$	$2.56e-2$
8	416	426	10
9	$-5.12e-2$	$+5.12e-2$	$1.024e-1$
10	-0.32	$+0.32$	0.64
11	-6	$+6$	12
12	0.5	1.5	1
13	0.5	1.5	1

Tabela 6 – Os limites dos espaços de busca e as constantes gravitacionais normalizadas da primeira série de experimentos

Foram usados os seguintes argumentos: número de partículas $P = 50$, número de iterações $T = 1000$, número de dimensões $D = 30$ e $\alpha = 20$. Foram realizadas 30 execuções de cada algoritmo. A média aritmética da convergência da função 8 e as médias geométricas das convergências das outras funções são mostradas nas figuras 17 e 18. Figuras mais detalhadas são mostradas no Apêndice F. Só as 100 e 250 primeiras iterações das funções 6 e 8 são mostradas nos gráficos, respectivamente. A média aritmética e o intervalo de confiança de 95% da solução de cada algoritmo são mostrados na Tabela 7.

Função	*SGSA* μ	*SGSA* γ	*GSA-NGC* μ	*GSA-NGC* γ
1	$2.2971e-17$	$1.8208e-18$	$\mathbf{8.5403e-21}$	$\mathbf{7.2248e-22}$
2	$2.2995e-8$	$1.1703e-9$	$\mathbf{4.6989e-11}$	$\mathbf{3.5193e-12}$
3	$5.8285e-2$	$7.5958e-3$	$\mathbf{2.6571e-2}$	$\mathbf{2.8575e-03}$
4	$3.2929e-9$	$3.3416e-10$	$\mathbf{6.7039e-11}$	$\mathbf{4.1917e-12}$
5	$1.6842e-3$	$\mathbf{6.6000e-4}$	$\mathbf{2.9855e-4}$	$1.1259e-3$
6	**0**	0	**0**	0
7	$2.7471e-5$	$9.0276e-6$	$\mathbf{1.898e-5}$	$\mathbf{8.2349e-6}$
8	$\mathbf{-1.2569e+4}$	$1.9861e-12$	$\mathbf{-1.2569e+4}$	$1.9861e-12$
9	$7.1054e-15$	$9.5894e-16$	**0**	**0**
10	$3.6115e-9$	$2.0937e-10$	$\mathbf{2.2837e-11}$	$\mathbf{1.3195e-12}$
11	**0**	0	**0**	0
12	**5.5363**	$\mathbf{2.8562e-2}$	6.7629	$7.1314e-2$
13	$2.1096e-18$	$1.7595e-19$	$\mathbf{2.0597e-22}$	$\mathbf{1.7802e-23}$

Tabela 7 – Média (μ) e intervalo de confiança de 95% (γ) das funções de referência avaliadas em um espaço de busca pequeno

O *GSA-NGC* foi melhor do que o *SGSA* em sete funções: nas funções 1, 2, 3, 4, 10 e 13, convergindo em soluções até quatro ordens de magnitude melhores; e na função 9, convergindo em uma solução aproximadamente exata. Os dois foram estatisticamente iguais em cinco funções: 5, 6, 7, 8 e 11. A solução do *SGSA* foi boa em todas essas funções, mas a convergência foi relativamente pequena nas primeiras iterações até nas funções 6, 8 e 11, nas quais os dois convergiram em uma solução aproximadamente exata.

Na função 6, o *GSA-NGC* convergiu em exatamente três iterações em todos os experimentos, enquanto o *SGSA* convergiu em aproximadamente 100 iterações. Na função 8, os dois algoritmos convergiram em uma solução aproximadamente exata, porque o ótimo dessa função não fica nas extremidades do espaço de busca, nesse espaço modificado. Na função 12, o *SGSA* foi melhor do que o *GSA-NGC*, mas os dois não convergiram.

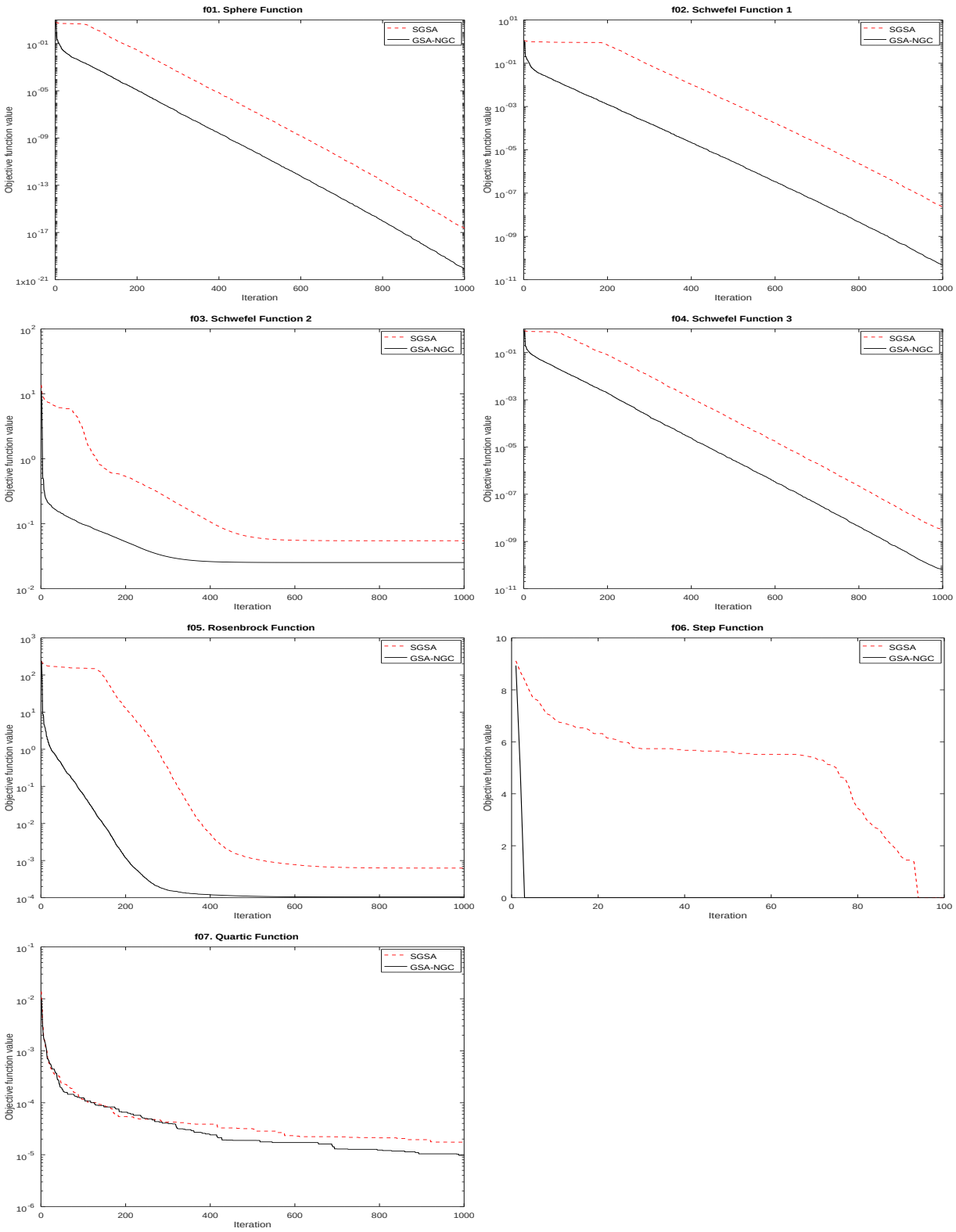

Figura 17 – As funções unimodais avaliadas em um espaço de busca relativamente pequeno

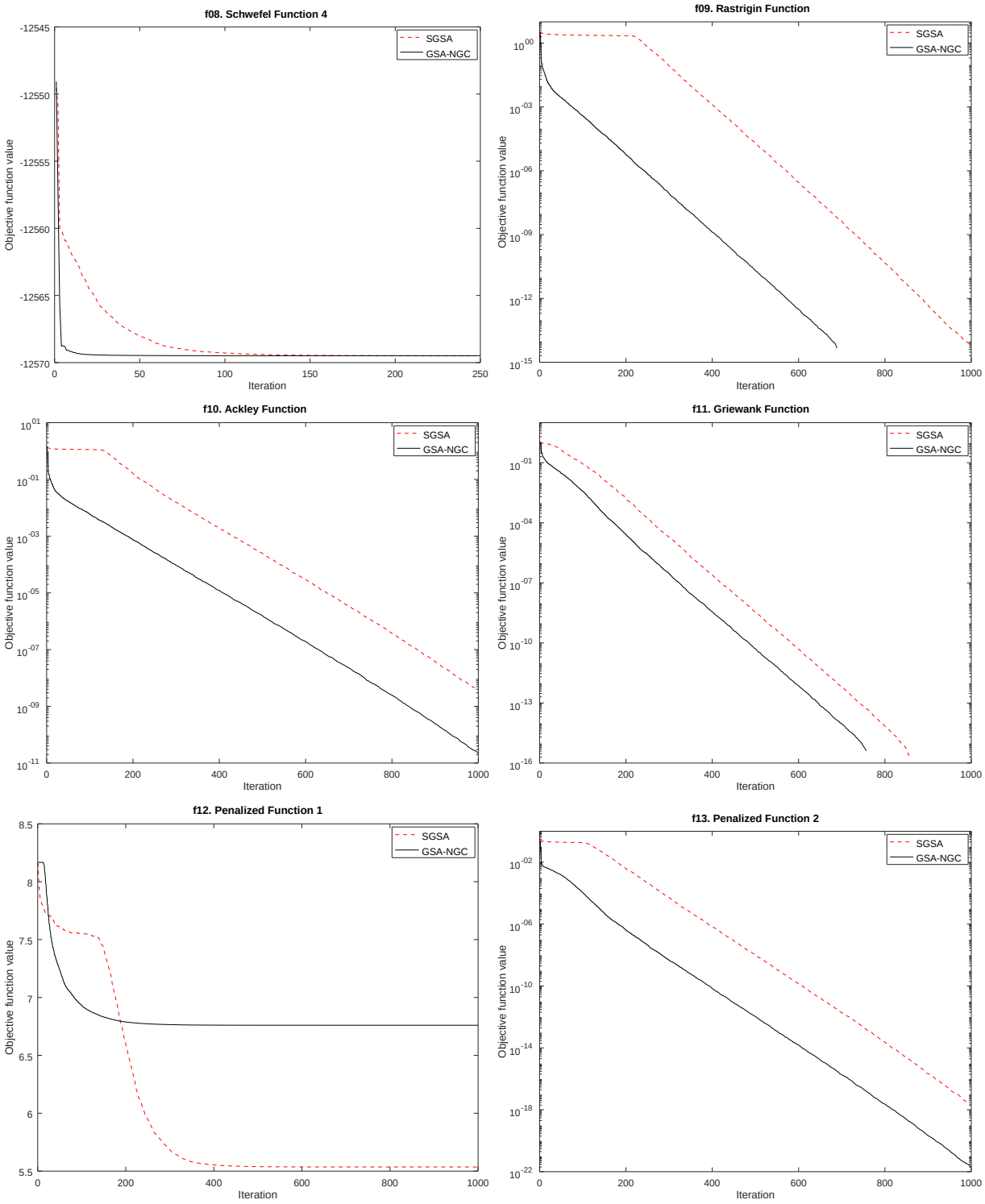

Figura 18 – As funções multimodais avaliadas em um espaço de busca relativamente pequeno

5.2 Experimentos usando um espaço de busca grande

Na segunda série de experimentos, as funções de referência são avaliadas em um espaço de busca relativamente grande, onde cada dimensão é 100 vezes maior do que a original (Tabela 2). Os limites do espaço de busca de cada função e as constantes gravitacionais normalizadas do *GSA-NGC* são mostradas na Tabela 8. O objetivo desses experimentos é demonstrar que a *NGC* também é apropriada para problemas com espaços de busca muito grandes.

Função	Limite Mínimo	Limite Máximo	G_0
1	-10000	$+10000$	20000
2	-1000	$+1000$	2000
3	-10000	$+10000$	20000
4	-10000	$+10000$	20000
5	-3000	$+3000$	6000
6	-10000	$+10000$	20000
7	-128	$+128$	256
8	-50000	$+50000$	100000
9	-512	$+512$	1024
10	-3200	$+3200$	6400
11	-60000	$+60000$	120000
12	-5000	$+5000$	10000
13	-5000	$+5000$	10000

Tabela 8 – Os limites dos espaços de busca e as constantes gravitacionais normalizadas da segunda série de experimentos

Foram usados os seguintes argumentos: número de partículas $P = 50$, número de iterações $T = 1000$, número de dimensões $D = 30$ e $\alpha = 20$. A média aritmética da convergência da função 8 e as médias geométricas das convergências das outras funções são mostradas nas figuras 19 e 20. Figuras mais detalhadas são mostradas no Apêndice G. Só as 500 e 250 primeiras iterações das funções 6 e 8 são mostradas nos gráficos, respectivamente. A média aritmética e o intervalo de confiança de 95% da solução de cada algoritmo são mostrados na Tabela 9.

O *GSA-NGC* foi melhor do que o *SGSA* em 12 funções: nas funções 1, 2, 3, 4, 5, 7, 9, 10, 11, 12, 13, convergindo em soluções até 21 ordens de magnitude melhores; e na função 6, convergindo em uma solução ótima exata. Os dois ficaram empatados na função 8, considerando o intervalo de confiança. Esses resultados tambem mostram que o *GSA* é muito sensível à constante gravitacional inicial e que uma constante muito pequena e pior do que uma muito grande.

Na função 6, o *GSA-NGC* convergiu em uma solução ótima exata em aproximadamente 350 iterações, enquanto o *SGSA* não convergiu.

Função	*SGSA* μ	*SGSA* γ	*GSA-NGC* μ	*GSA-NGC* γ
1	$5.1286e+8$	$1.8719e+7$	$\mathbf{9.2256e-13}$	$\mathbf{8.2629e-14}$
2	$1.8747e+3$	$\mathbf{9.0072e+1}$	$\mathbf{8.7585e+2}$	$1.0569e+2$
3	$1.1097e+9$	$1.0155e+8$	$\mathbf{2.7262e+6}$	$\mathbf{3.9477e+5}$
4	$7.7006e+3$	$1.0204e+2$	$\mathbf{5.9826e-7}$	$\mathbf{3.8500e-8}$
5	$3.2424e+15$	$2.4717e+14$	$\mathbf{5.9511e+3}$	$\mathbf{3.7620e+3}$
6	$4.9287e+8$	$2.1894e+7$	$\mathbf{0}$	$\mathbf{0}$
7	$4.4117e-2$	$1.0665e-2$	$\mathbf{2.7421e-2}$	$\mathbf{4.4733e-3}$
8	$\mathbf{-2.9766e+5}$	$\mathbf{1.5830e+4}$	$-2.6435e+5$	$1.8250e+4$
9	$6.0513e+3$	$1.1376e+3$	$\mathbf{1.8307e+1}$	$\mathbf{1.8169}$
10	$2.1087e+1$	$1.6165e-2$	$\mathbf{2.0000e+1}$	$\mathbf{1.5015e-14}$
11	$5.5834e+6$	$2.1637e+5$	$\mathbf{5.7536e-4}$	$\mathbf{7.9208e-4}$
12	$6.3745e+16$	$4.4064e+15$	$\mathbf{3.4556e-3}$	$\mathbf{6.7728e-3}$
13	$6.5013e+16$	$4.2851e+15$	$\mathbf{3.6625e-4}$	$\mathbf{7.1782e-4}$

Tabela 9 – Média (μ) e intervalo de confiança de 95% (γ) das funções de referência avaliadas em um espaço de busca relativamente grande

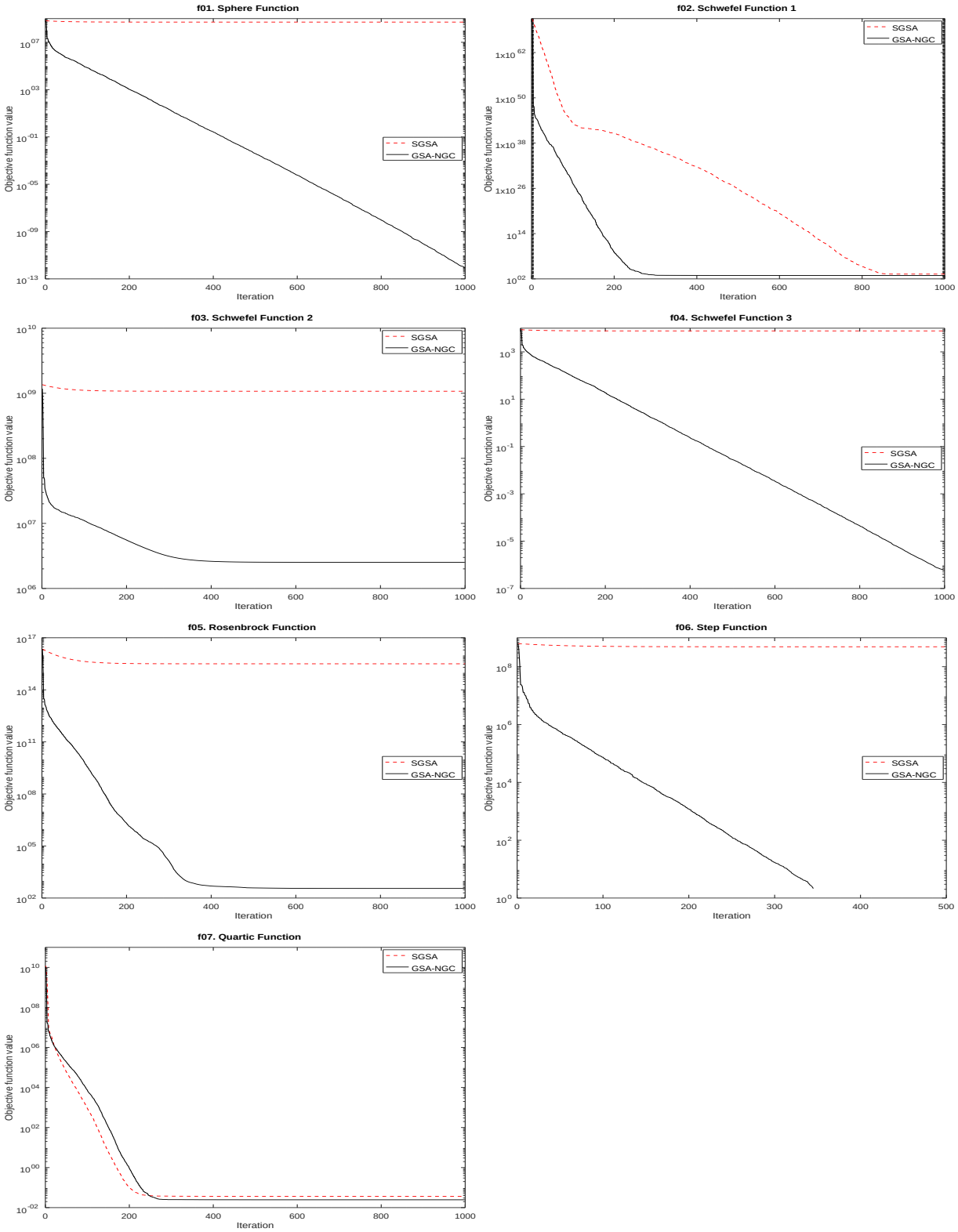

Figura 19 – As funções unimodais avaliadas em um espaço de busca relativamente grande

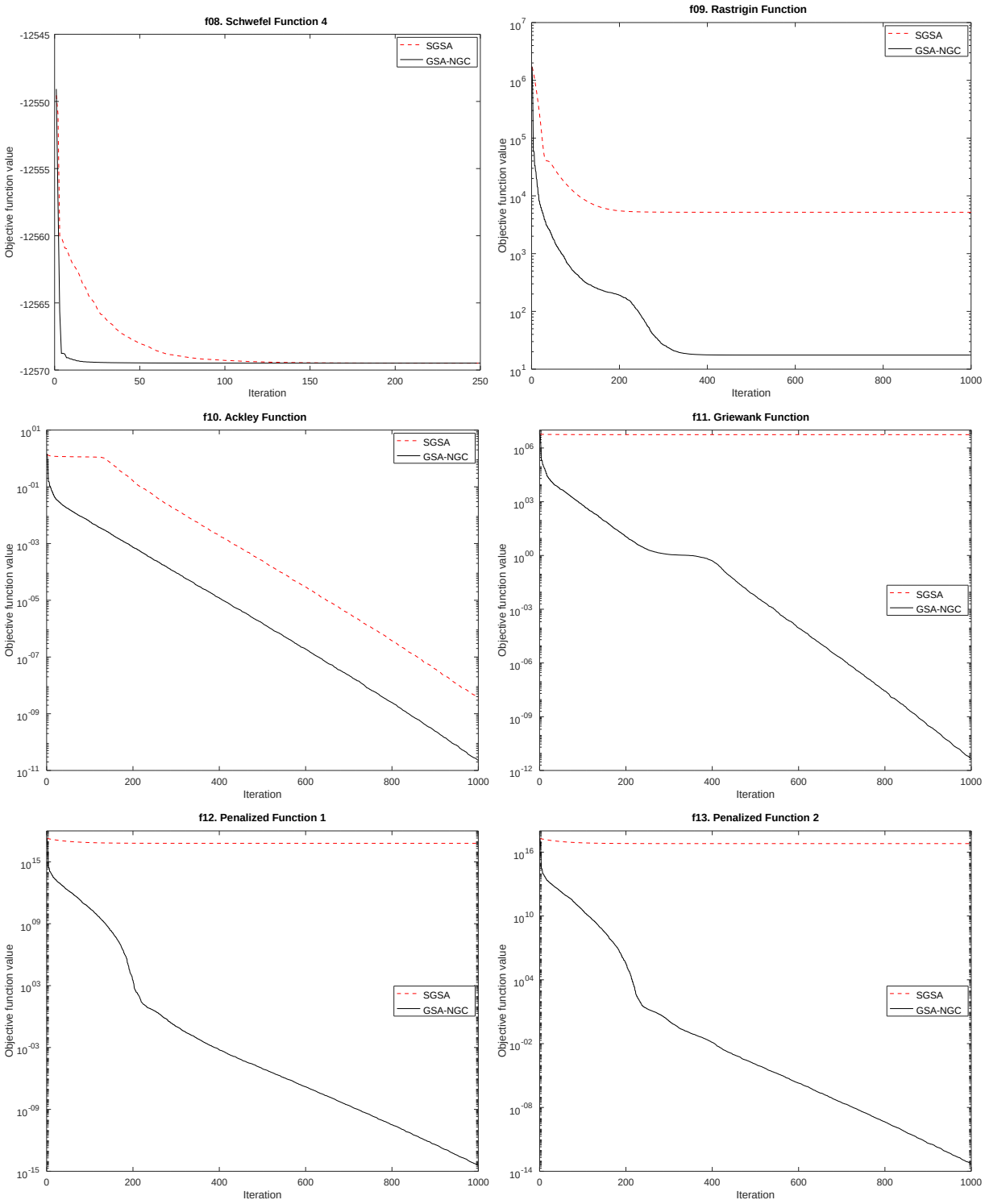

Figura 20 – As funções multimodais avaliadas em um espaço de busca relativamente grande

5.3 Experimentos usando um espaço de busca irregular

Na terceira série de experimentos, as funções são avaliadas em um espaço de busca grandemente irregular de dimensão 11, cada dimensão tendo uma ordem de magnitude maior do que a outra, de 10^{-5} a 10^{+5} vezes a original (Tabela 2). Os limites do espaço de busca de cada função e as constantes gravitacionais normalizadas do *GSA-NGC* são mostradas na Tabela 10. O objetivo desses experimentos é demonstrar que a *NGC* é apropriada até para os espaços de busca mais irregulares.

	Dimensão do espaço de busca					
Função	1	2	3	...	11	G_0
1	$\pm 1e-3$	$\pm 1e-2$	$\pm 1e-1$	...	$\pm 1e+7$	$2.0202e+6$
2	$\pm 1e-4$	$\pm 1e-3$	$\pm 1e-2$	...	$\pm 1e+6$	$2.0202e+5$
3	$\pm 1e-3$	$\pm 1e-2$	$\pm 1e-1$	...	$\pm 1e+7$	$2.0202e+6$
4	$\pm 1e-3$	$\pm 1e-2$	$\pm 1e-1$	...	$\pm 1e+7$	$2.0202e+6$
5	$1 \pm 3e-4$	$1 \pm 3e-3$	$1 \pm 3e-2$	...	$1 \pm 3e+6$	$6.0606e+5$
6	$\pm 1e-3$	$\pm 1e-2$	$\pm 1e-1$	...	$\pm 1e+7$	$2.0202e+6$
7	$\pm 1.28e-5$	$\pm 1.28e-4$	$\pm 1.28e-3$	...	$\pm 1.28e+5$	$2.5859e+4$
8	$421 \pm 5e-3$	$421 \pm 5e-2$	$421 \pm 5e-1$	...	$421 \pm 5e+7$	$1.0101e+7$
9	$\pm 5.12e-5$	$\pm 5.12e-4$	$\pm 5.12e-3$	...	$\pm 5.12e+5$	$1.0343e+5$
10	$\pm 3.2e-4$	$\pm 3.2e-3$	$\pm 3.2e-2$	...	$\pm 3.2e+6$	$6.4646e+5$
11	$\pm 6e-3$	$\pm 6e-2$	$\pm 6e-1$	...	$\pm 6e+7$	$1.2121e+7$
12	$1 \pm 5e-4$	$1 \pm 5e-3$	$1 \pm 5e-2$	...	$1 \pm 5e+6$	$1.0101e+6$
13	$1 \pm 5e-4$	$1 \pm 5e-3$	$1 \pm 5e-2$	...	$1 \pm 5e+6$	$1.0101e+6$

Tabela 10 – Os limites dos espaços de busca e as constantes gravitacionais normalizadas da terceira série de experimentos

Foram usados os seguintes argumentos: número de partículas $P = 50$, número de iterações $T = 1000$ e $\alpha = 20$. A média aritmética da convergência da função 8 e as médias geométricas das convergências das outras funções são mostradas nas figuras 21 e 22. Figuras mais detalhadas são mostradas no Apêndice H. Só as 250 primeiras iterações da função 8 são mostradas no gráfico. A média aritmética e o intervalo de confiança de 95% da solução de cada algoritmo são mostrados na Tabela 11.

O *GSA-NGC* foi melhor do que o *SGSA* em 12 funções: nas funções 1, 2, 3, 4, 5, 7, 9, 10, 11, 12 e 13, convergindo em soluções até 34 ordens de magnitude melhores; e na função 6, convergindo em uma solução ótima exata. Os dois ficaram empatados na função 8, considerando o intervalo de confiança. Os resultados da terceira série de experimentos são semelhantes aos da segunda. Essa semelhança aponta para a conclusão de que os espaços de busca irregulares da terceira série de experimentos também são relativamente grandes, e a constante gravitacional original é muito pequena.

Na função 6, o *GSA-NGC* convergiu em uma solução ótima exata em aproximadamente 500 iterações, enquanto o *SGSA* não convergiu.

Função	*SGSA* μ	*SGSA* γ	*GSA-NGC* μ	*GSA-NGC* γ
1	$3.3103e+11$	$1.3314e+11$	$\mathbf{7.7405e-10}$	$\mathbf{8.9363e-11}$
2	$9.9866e+4$	$2.9211e+4$	$\mathbf{9.4045e-6}$	$\mathbf{6.0958e-7}$
3	$3.4015e+11$	$1.3379e+11$	$\mathbf{2.2029e-9}$	$\mathbf{3.1197e-10}$
4	$3.5038e+5$	$5.8932e+4$	$\mathbf{2.0126e-5}$	$\mathbf{1.6776e-6}$
5	$4.8769e+18$	$2.5670e+18$	$\mathbf{7.7783e+12}$	$\mathbf{4.9965e+12}$
6	$2.1835e+11$	$5.9126e+10$	$\mathbf{0}$	$\mathbf{0}$
7	$2.0982e+16$	$1.6527e+16$	$\mathbf{4.4300e-3}$	$\mathbf{7.0931e-4}$
8	$\mathbf{-4.5291e+7}$	$1.6195e+6$	$-4.4514e+7$	$\mathbf{1.2426e+6}$
9	$8.0720e+8$	$2.9013e+8$	$\mathbf{2.7522}$	$\mathbf{6.5245e-1}$
10	$2.0260e+1$	$2.7021e-2$	$\mathbf{2.0000e+1}$	$\mathbf{6.2844e-11}$
11	$1.8691e+9$	$5.9484e+8$	$\mathbf{1.3106e-3}$	$\mathbf{2.5688e-3}$
12	$3.4391e+23$	$2.0862e+23$	$\mathbf{5.1149}$	$\mathbf{9.8184e-1}$
13	$5.7623e+23$	$3.3813e+23$	$\mathbf{2.0525e-11}$	$\mathbf{3.1333e-12}$

Tabela 11 – Média (μ) e intervalo de confiança de 95% (γ) das funções de referência avaliadas em um espaço de busca irregular

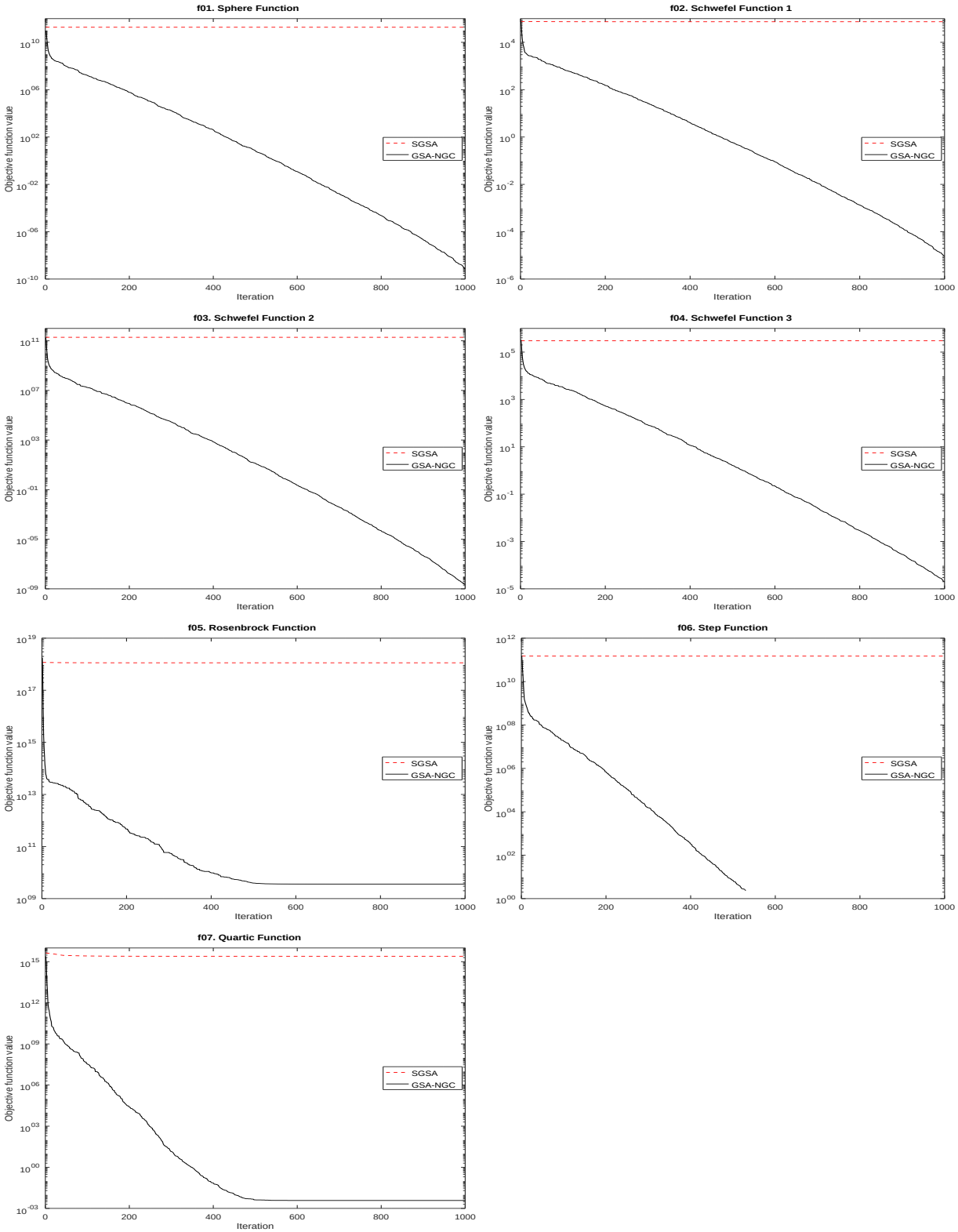

Figura 21 – As funções unimodais avaliadas em um espaço de busca irregular

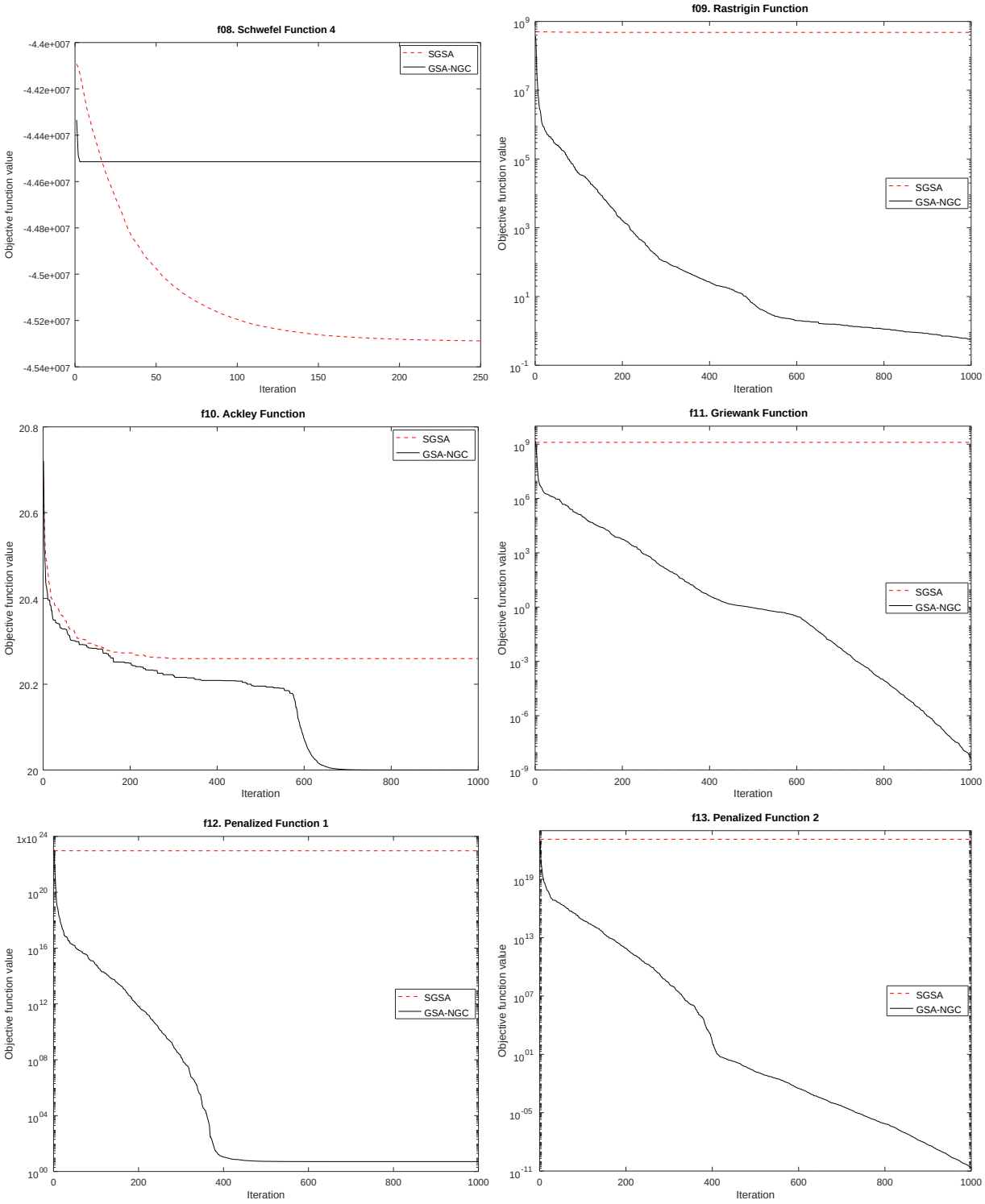

Figura 22 – As funções multimodais avaliadas em um espaço de busca irregular

5.4 Considerações Finais

Foram realizadas três séries de experimentos: a primeira, usando um espaço de busca pequeno; a segunda, usando um espaço de busca grande; e a terceira, usando um espaço de busca irregular. O *GSA-NGC* foi melhor do que o *SGSA* em sete funções na primeira, e 12 na segunda e na terceira; com soluções até quatro, 21 e 34 ordens de magnitude melhores, respectivamente. O *GSA-NGC* só foi pior do que o *SGSA* em uma função, na primeira série, com a solução na mesma ordem de magnitude. Os resultados assim mostram que a *NGC* é apropriada para funções unimodais e multimodais com qualquer espaço de busca, significando uma melhora na versatilidade do *GSA*, que é muito sensível à constante gravitacional inicial.

6 Conclusão

A heurística proposta para determinar uma constante gravitacional do *GSA*, proporcional ao espaço de busca do problema: a *Normalized Gravitational Constant (NGC)*, aumentou a robustez e a adaptabilidade do algoritmo para espaços de busca irregulares em comparação ao GSA original. Essa heurística é baseada na Teoria de Brans-Dicke, que diz que a constante gravitacional é diretamente proporcional ao raio do universo visível e inversamente proporcional à massa; e justificada pela observação de que a constante gravitacional inicial do *GSA* deve ser proporcional ao espaço de busca do problema, observação essa que foi confirmada experimentalmente.

Foram realizadas cinco séries de experimentos, usando treze funções de referência conhecidas: sete unimodais e seis multimodais, variando a constante gravitacional inicial do *GSA* em várias ordens de magnitude. O objetivo desses experimentos era determinar a relação entre a constante gravitacional inicial e o tamanho do espaço de busca. Os resultados confirmaram que, se a constante gravitacional inicial for muito grande, a convergência do algoritmo vai ser muito pequena durante as primeiras iterações da execução; mas, se ela for muito pequena, a convergência vai ser muito pequena durante toda a execução. As melhores soluções foram obtidas pelas constantes mais próximas do tamanho do espaço de busca.

Usando a heurística proposta, o *Gravitational Search Algorithm with Normalized Gravitational Constant (GSA-NGC)* foi comparado com o *Stardard GSA (SGSA)*, usando as mesmas funções de referência. Foram realizadas três séries de experimentos, variando o tamanho do espaço de busca:

1. Na primeira série de experimentos, as funções de referência foram avaliadas em um espaço de busca pequeno. O *GSA-NGC* foi melhor do que o *SGSA* em sete funções, convergindo em soluções até quatro ordens de magnitude melhores.

2. Na segunda série de experimentos, as funções de referência foram avaliadas em um espaço de busca grande. O *GSA-NGC* foi melhor do que o *SGSA* em 11 funções, convergindo em soluções até 21 ordens de magnitude melhores.

3. Na terceira série de experimentos, as funções referência foram avaliadas em um espaço de busca irregular. O *GSA-NGC* foi melhor do que o *SGSA* em 12 funções, convergindo em soluções até 34 ordens de magnitude melhores.

Resumindo, os resultados confirmaram que a *NGC* é apropriada para funções unimodais e multimodais, significando uma melhora na versatilidade do GSA, destacando sua melhoria significativa em espaço de busca extremamente irregulares.

Contudo, a constante gravitacional normalizada é baseada em uma técnica heurística, logo seu valor não é ótimo, mas sub-ótimo, com resultados bons para os problemas. Aliás, a constante ótima depende da função objetiva e, portanto, não pode ser determinada a priori. É evidente que uma constante gravitacional inicial tão boa quanto a normalizada, ou melhor, poderia ser determinada por experimentos. No entanto, a heurística proposta é uma operação simples e rápida, enquanto essas tentativas e correções tomariam horas de cálculo.

6.1 Trabalhos futuros

Três sugestões de trabalhos futuros para esta proposta são apontadas:

1. A *NGC* pode ser combinada com outras versões do *GSA*, que também são muito sensíveis à constante gravitacional inicial.
2. Uma constante gravitacional vetorial, cada dimensão da constante gravitacional sendo proporcional a uma dimensão do espaço de busca do problema, também é cogitada.
3. Utilizar a proposta de parâmetros iniciais para outros algoritmos de otimização global sensíveis a variações no espaço de busca.

APÊNDICE A – Implementação do *GSA* em Octave

Código A.1 – initialization.m

```
function positions=initialization(
    particles, dimensions, minLimits, maxLimits)
    positions=rand(particles, dimensions).*...
        (maxLimits-minLimits)+minLimits;
end
```

Código A.2 – getObjectiveValues.m

```
function objectiveValues=getObjectiveValues(
    objectiveFunction, positions);
    persistent particles=size(positions, 1);
    objectiveValues=zeros(1, particles);
    for i=1:particles
        objectiveValues(i)=feval(
            objectiveFunction, positions(i,:));
    end
end
```

Código A.3 – getSolution.m

```
function [solutionValue, solutionArguments]=getSolution(
    solutionValue, solutionArguments, objectiveValues,
    positions, minimization)
    if minimization
        [solutionAux, index]=min(objectiveValues);
        if solutionAux<solutionValue
            solutionValue=solutionAux;
            solutionArguments=positions(index,:);
        end
    else
        [solutionAux, index]=max(objectiveValues);
        if solutionAux>solutionValue
            solutionValue=solutionAux;
            solutionArguments=positions(index,:);
        end
    end
end
```

Código A.4 – getNormalizedValues.m

```
function normalizedValues=getNormalizedValues(
    objectiveValues, minimization)
    maxValue=max(objectiveValues);
    minValue=min(objectiveValues);
    if maxValue>minValue
        if minimization
            normalizedValues=(objectiveValues-maxValue)/...
                (minValue-maxValue);
        else
            normalizedValues=(objectiveValues-minValue)/...
                (maxValue-minValue);
        end

    else
        persistent particles=size(objectiveValues, 2);
        normalizedValues=ones(1, particles);
    end
end
```

Código A.5 – getActiveParticles.m

```
function activeParticles=getActiveParticles(
    particles, strategy, iteration, maxIteration)

    if strategy
        inactiveParticles=(particles-strategy)*...
            (iteration/maxIteration);
        activeParticles=particles-round(inactiveParticles);
    else
        activeParticles=0;
    end
end
```

Código A.6 – getGConstant.m

```
function gConstant=getGConstant(
    gConstant0, alpha, iteration, maxIteration)
    gConstant=gConstant0*exp(-alpha*iteration/maxIteration);
end
```

Código A.7 – getGForce.m

```
function gForce=getGForce(
    positions, masses, activeParticles, gConstant)

    persistent particles=size(positions, 1);
    persistent dimensions=size(positions, 2);
    gForce=zeros(particles, dimensions);

    if activeParticles
        [sortedValues indexes]=sort(masses, "descend");
        indexes=indexes(1:activeParticles);
        for i=1:particles
            for j=indexes
                if j~=i
                    difference=positions(j,:)-positions(i,:);
                    gForce(i,:)=gForce(i,:)+rand(1, dimensions).*...
                    (masses(j)*difference)/(norm(difference)+eps);
                end
            end
        end

    else
        for i=1:particles
            for j=1:particles
                if j~=i
                    difference=positions(j,:)-positions(i,:);
                    gForce(i,:)=gForce(i,:)+rand(1, dimensions).*...
                    (masses(j)*difference)/(norm(difference)+eps);
                end
            end
        end
    end

    gForce=gConstant*gForce;
end
```

Código A.8 – getPositions.m

```
function [positions velocities]=getPositions(
    positions, velocities, gForce)
    persistent particles=size(positions, 1);
    persistent dimensions=size(positions, 2);
    velocities=rand(particles, dimensions).*...
        velocities+gForce;
    positions=positions+velocities;
end
```

Código A.9 – spaceLimits.m

```
function positions=spaceLimits(
    positions, minLimits, maxLimits)
    persistent particles=size(positions, 1);
    persistent dimensions=size(positions, 2);
    for i=1:particles
        aux=positions(i,:)<minLimits |...
            positions(i,:)>maxLimits;
        positions(i,:)=(rand(1, dimensions).*...
            (maxLimits-minLimits)+minLimits).*aux+...
            positions(i,:).*~aux;
    end
end
```

Código A.10 – gravitationalSearchAlgorithm.m

```
function [solutionValue, solutionArguments, solutionGraph]=...
    gravitationalSearchAlgorithm(
    objectiveFunction, minLimits, maxLimits, dimensions, minimization,
    particles, maxIterations, strategy, alpha, beta)

    positions=initialization(
        particles, dimensions, minLimits, maxLimits);
    velocities=zeros(particles, dimensions);
    solutionArguments=zeros(1, dimensions);
    solutionGraph=zeros(1, maxIterations);

    if minimization
        solutionValue=+Inf;
    else
        solutionValue=-Inf;
    end

    if beta
        gConstant0=mean(maxLimits-minLimits)*beta;
    else
        gConstant0=100;
    end

    for iteration=1:maxIterations
        objectiveValues=getObjectiveValues(
            objectiveFunction, positions);
        [solutionValue, solutionArguments]=getSolution(
            solutionValue, solutionArguments, objectiveValues,
            positions, minimization);
        solutionGraph(iteration)=solutionValue;

        normalizedValues=getNormalizedValues(
            objectiveValues, minimization);
        activeParticles=getActiveParticles(
            particles, strategy, iteration, maxIterations);
        masses=normalizedValues/sum(normalizedValues);

        gConstant=getGConstant(
            gConstant0, alpha, iteration, maxIterations);
        gForce=getGForce(
            positions, masses, activeParticles, gConstant);
        [positions velocities]=getPositions(
            positions, velocities, gForce);
        positions=spaceLimits(
            positions, minLimits, maxLimits);
    end

    clear functions;
end
```

APÊNDICE B – Funções de Referência

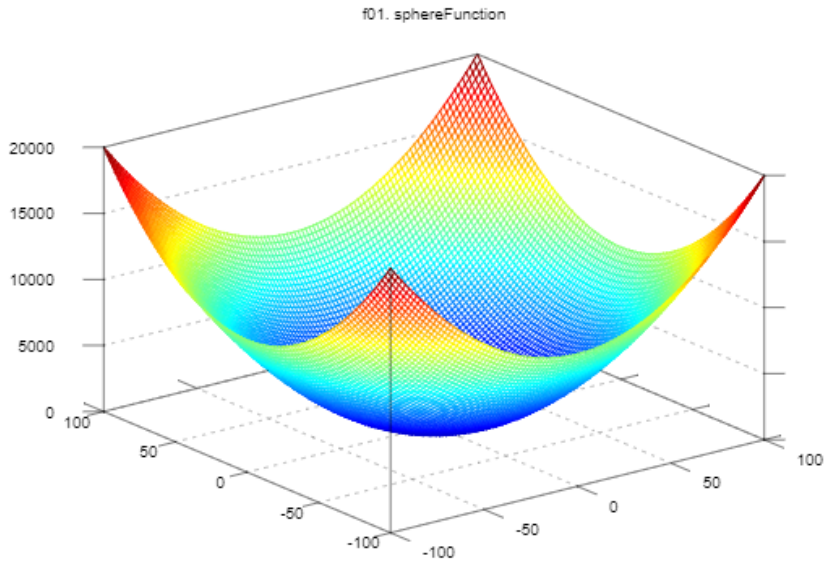

Figura 23 – Sphere Function: $\mathbb{R}^2 \rightarrow \mathbb{R}$

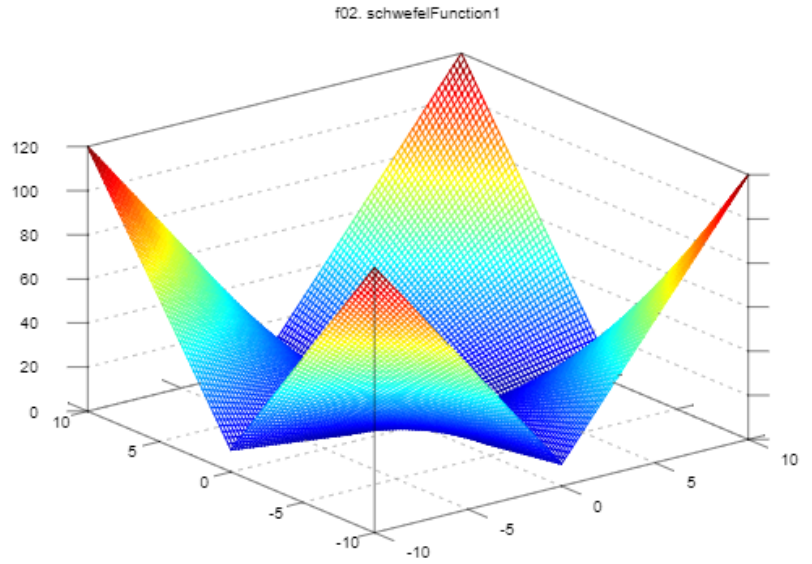

Figura 24 – Schwefel Function 1: $\mathbb{R}^2 \rightarrow \mathbb{R}$

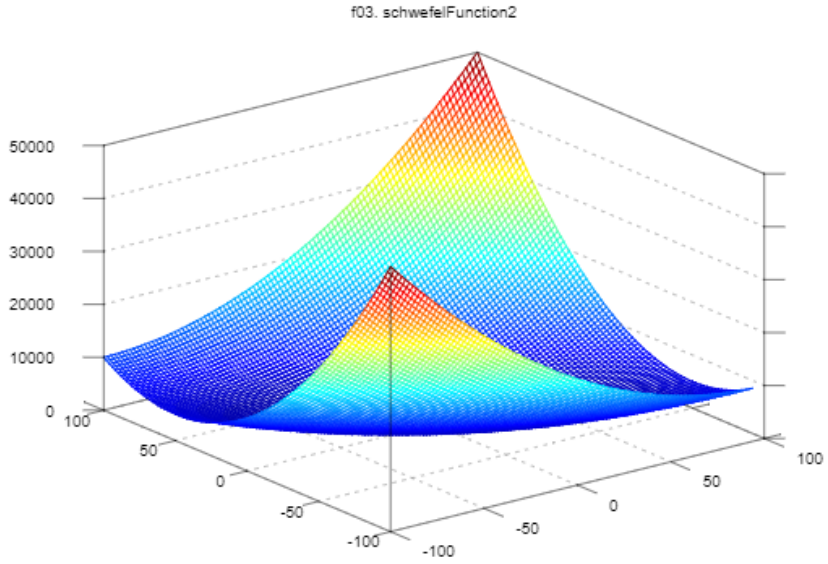

Figura 25 – Schwefel Function 2: $\mathbb{R}^2 \rightarrow \mathbb{R}$

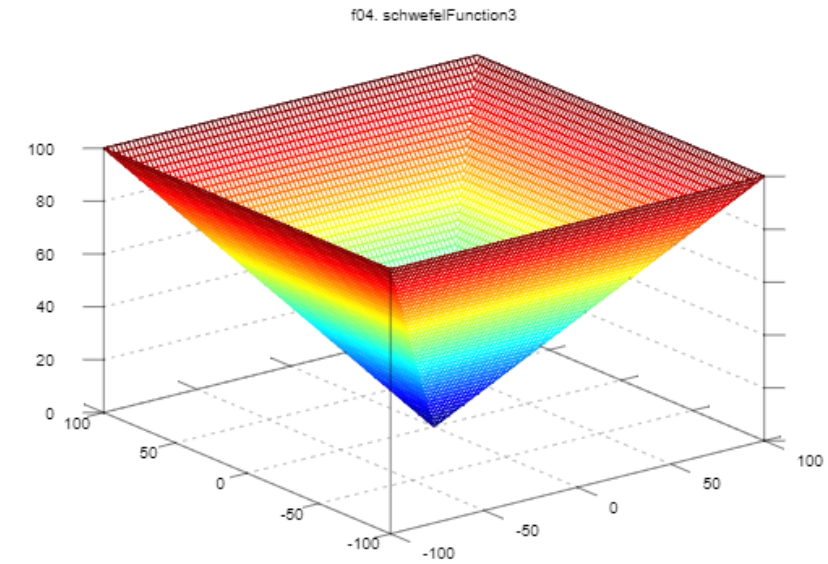

Figura 26 – Schwefel Function 3: $\mathbb{R}^2 \rightarrow \mathbb{R}$

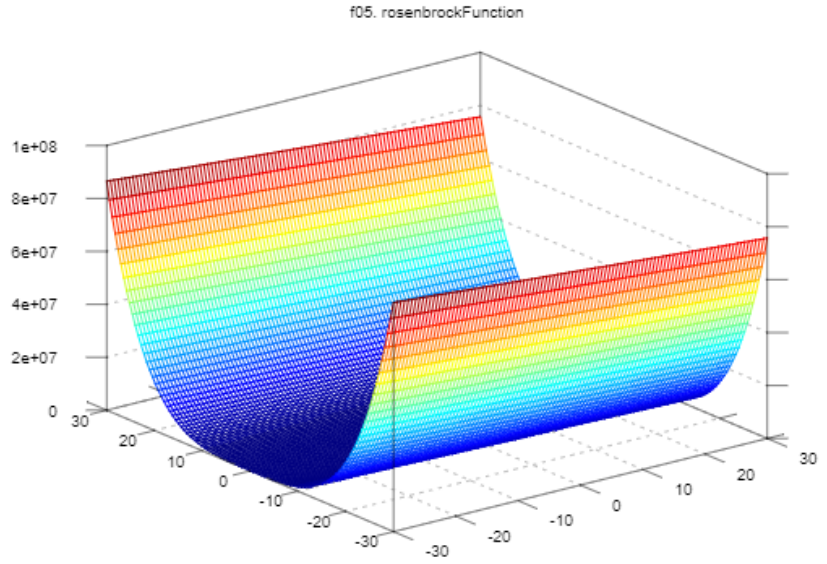

Figura 27 – Rosenbrock Function: $\mathbb{R}^2 \to \mathbb{R}$

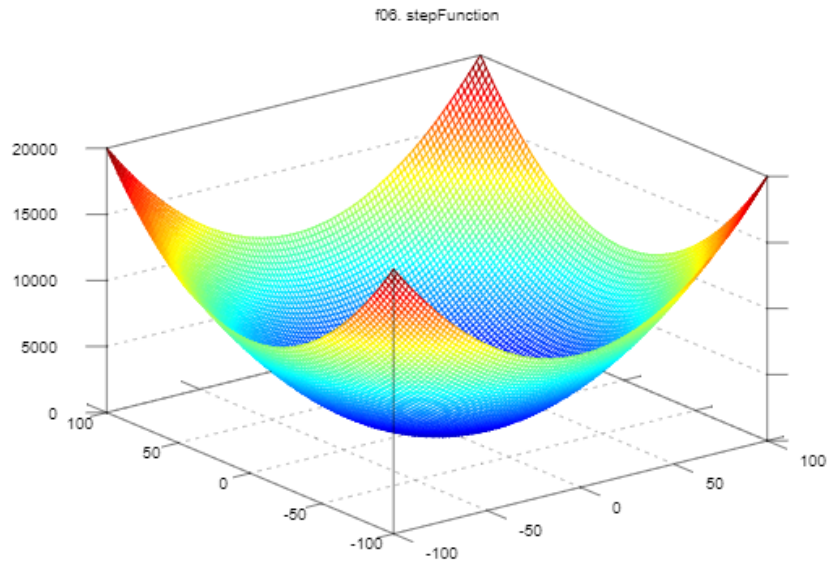

Figura 28 – Step Function: $\mathbb{R}^2 \to \mathbb{R}$

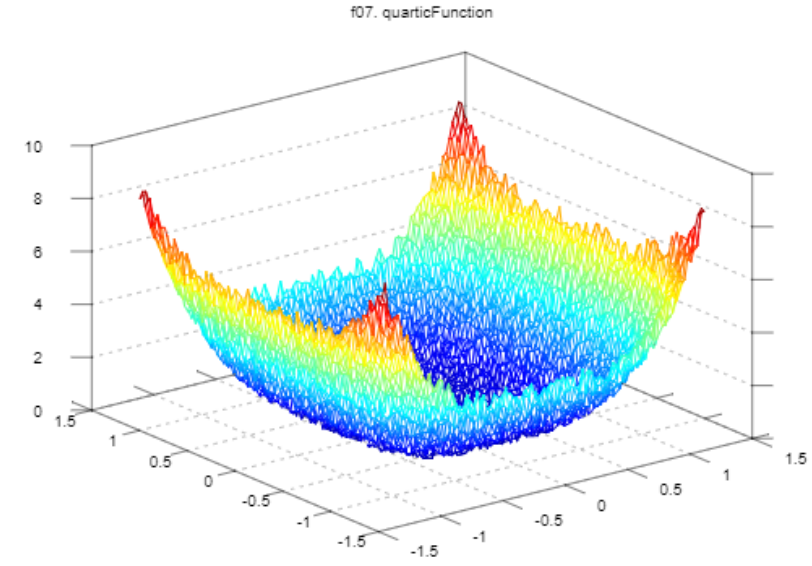

Figura 29 – Quartic Function: $\mathbb{R}^2 \rightarrow \mathbb{R}$

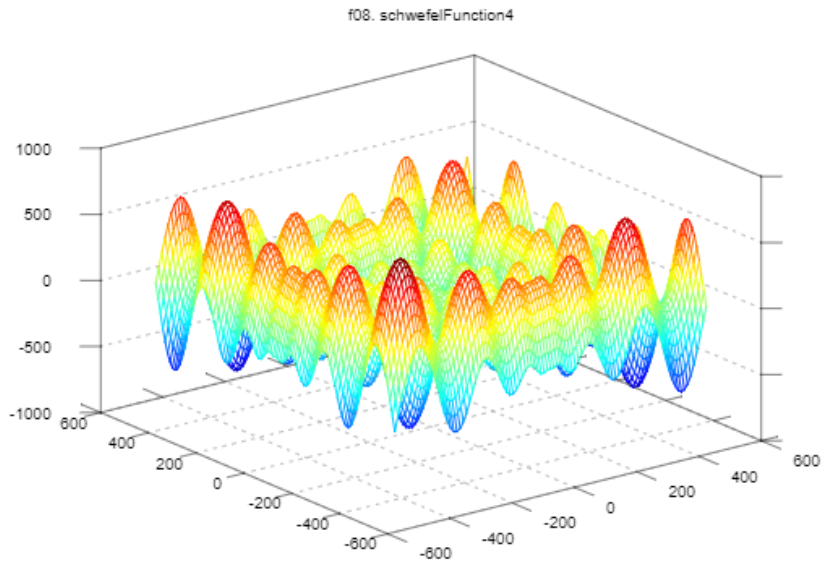

Figura 30 – Schwefel Function 4: $\mathbb{R}^2 \rightarrow \mathbb{R}$

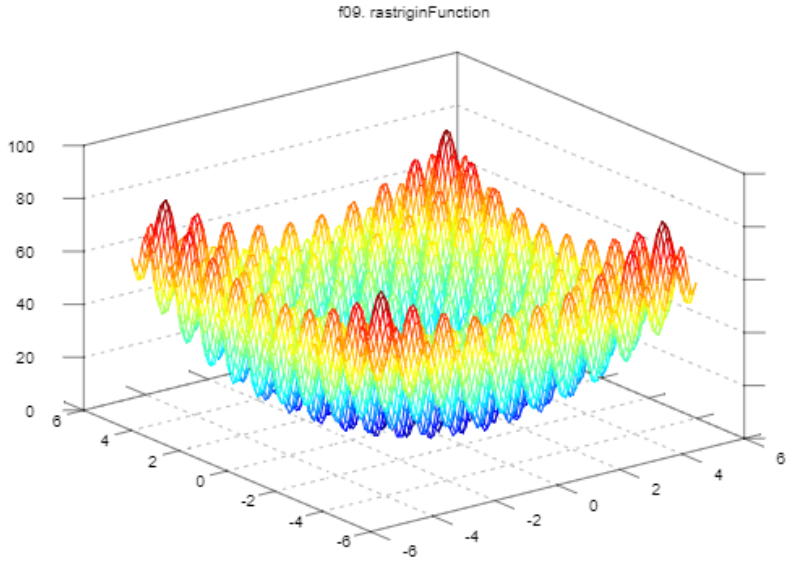

Figura 31 – Rastrigin Function: $\mathbb{R}^2 \rightarrow \mathbb{R}$

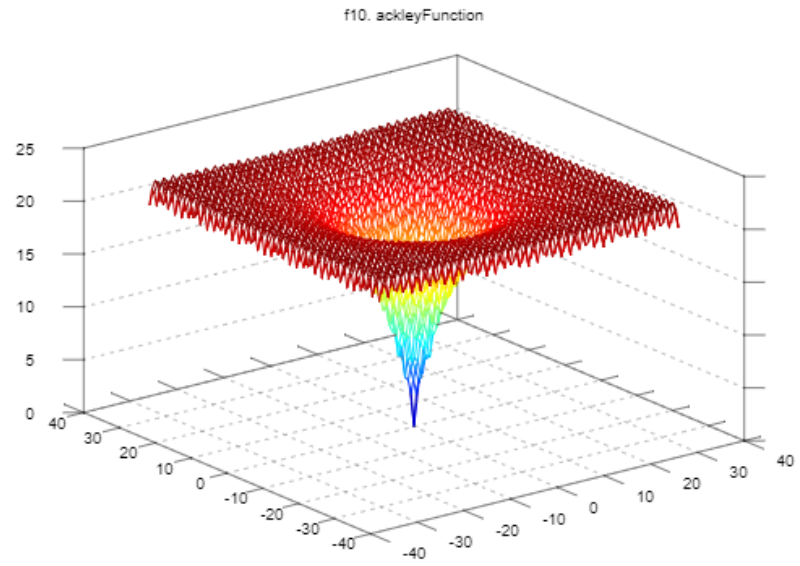

Figura 32 – Ackley Function: $\mathbb{R}^2 \rightarrow \mathbb{R}$

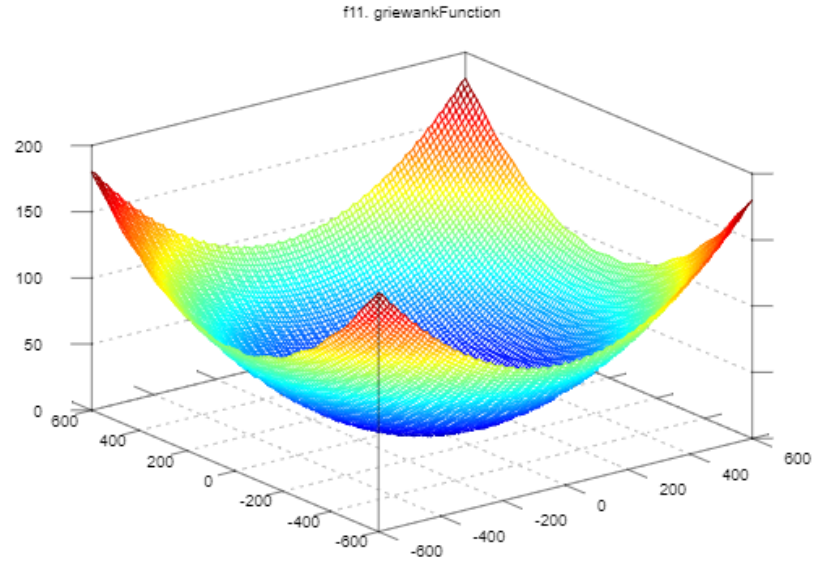

Figura 33 – Griewank Function: $\mathbb{R}^2 \rightarrow \mathbb{R}$

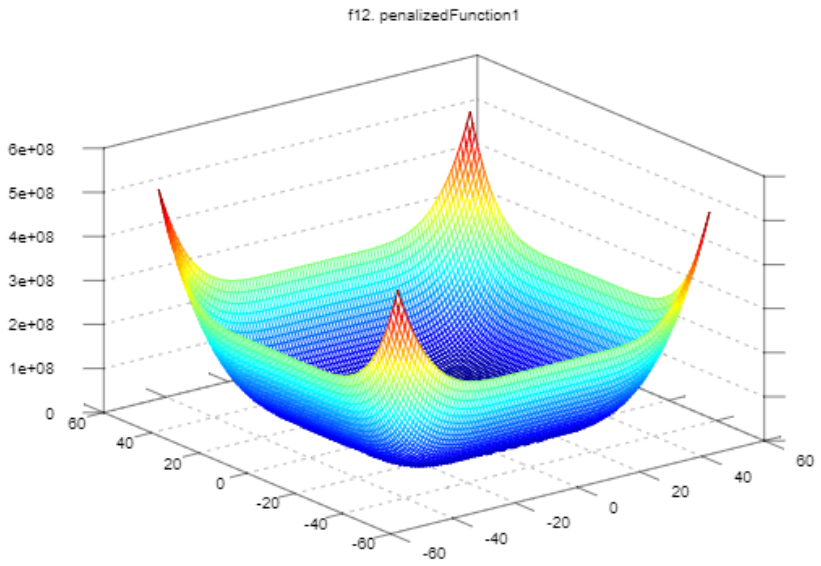

Figura 34 – Penalized Function 1: $\mathbb{R}^2 \rightarrow \mathbb{R}$

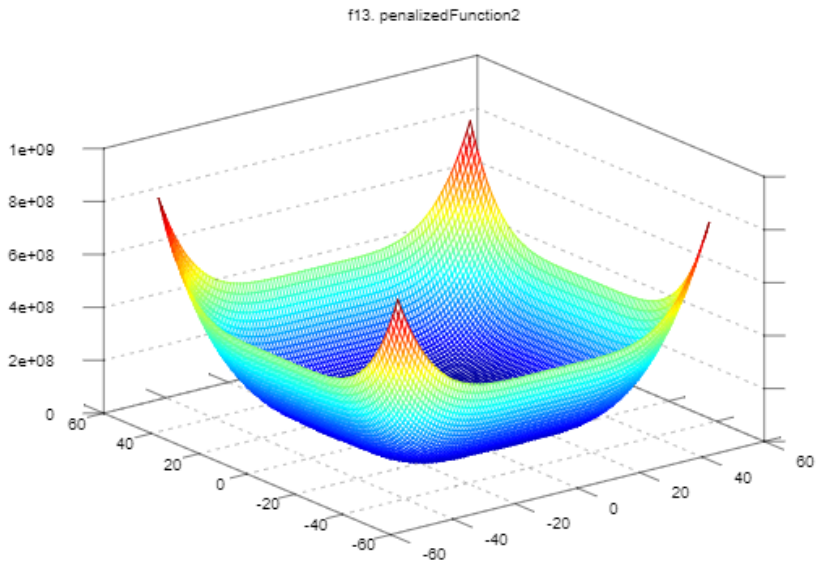

Figura 35 – Penalized Function 2: $\mathbb{R}^2 \rightarrow \mathbb{R}$

APÊNDICE C – Implementação das funções de referência em Octave

Código C.1 – sphereFunction.m

```
function value=sphereFunction(arguments)
    value=sum(arguments.^2);
end
```

Código C.2 – schwefelFunction1.m

```
function value=schwefelFunction1(arguments)
    value=sum(abs(arguments))+prod(abs(arguments));
end
```

Código C.3 – schwefelFunction2.m

```
function value=schwefelFunction2(arguments)
    value=sum(cumsum(arguments).^2);
end
```

Código C.4 – schwefelFunction3.m

```
function value=schwefelFunction3(arguments)
    value=max(abs(arguments));
end
```

Código C.5 – rosenbrockFunction.m

```
function value=rosenbrockFunction(arguments)
    value=sum(100*(arguments(2:end)-arguments(1:end-1).^2)+
        (arguments(1:end-1)-1).^2);
end
```

Código C.6 – stepFunction.m

```
function value=stepFunction(arguments)
    value=sum(floor(arguments+0.5).^2);
end
```

Código C.7 – quarticFunction.m

```
function value=quarticFunction(arguments)
    persistent aux=(1:size(arguments, 2));
    value=sum(aux.*arguments.^4)+rand;
end
```

Código C.8 – schwefelFunction4.m

```
function value=schwefelFunction4(arguments)
    value=-sum(arguments.*sin(sqrt(abs(arguments))));
end
```

Código C.9 – rastriginFunction.m

```
function value=rastriginFunction(arguments)
    value=sum(arguments.^2-10*cos(2*pi*arguments)+10);
end
```

Código C.10 – ackleyFunction.m

```
function value=ackleyFunction(arguments)
    persistent dimensions=size(arguments, 2);
    value=-20*exp(-0.2*sqrt(sum(arguments.^2)/dimensions))-...
        exp(sum(cos(2*pi*arguments))/dimensions)+e+20;
end
```

Código C.11 – griewankFunction.m

```
function value=griewankFunction(arguments)
    persistent aux=(1:size(arguments, 2));
    value=sum(arguments.^2)/4000-prod(cos(arguments./sqrt(aux)))+1;
end
```

Código C.12 – penalizedFunction1.m

```
function penalizedFunction1(arguments)
    persistent dimensions=size(arguments, 2);
    value=pi*(10*sin(pi*yFunction(arguments(1)))^2+...
        sum((yFunction(arguments(1:end-1))-1).^2.*...
        (10*sin(pi.*yFunction(arguments(2:end))).^2+1))+...
        (yFunction(arguments(end))-1)^2)/dimensions+...
        sum(uFunction(arguments, 10, 100, 4));
end
```

Código C.13 – penalizedFunction2.m

```
function value=penalizedFunction2(arguments)
    value=(sin(3*pi*arguments(1))^2+...
        sum((arguments(1:end-1)-1).^2.*...
        (sin(3*pi*arguments(2:end)).^2+1))+...
        (arguments(end)-1)^2.*...
        (sin(2*pi*arguments(end))^2+1))/10+...
        sum(uFunction(arguments, 5, 100, 4));
end
```

Código C.14 – yFunction.m

```
function value=yFunction(x)
    value=(x+1)/4+1;
end
```

Código C.15 – uFunction.m

```
function value=uFunction(x, a, k, m)
    value=(k.*(x-a).^m).*(x>a)+(k.*(-x-a).^m).*(x<-a);
end
```

APÊNDICE D – As funções de referência variando a constante gravitacional inicial

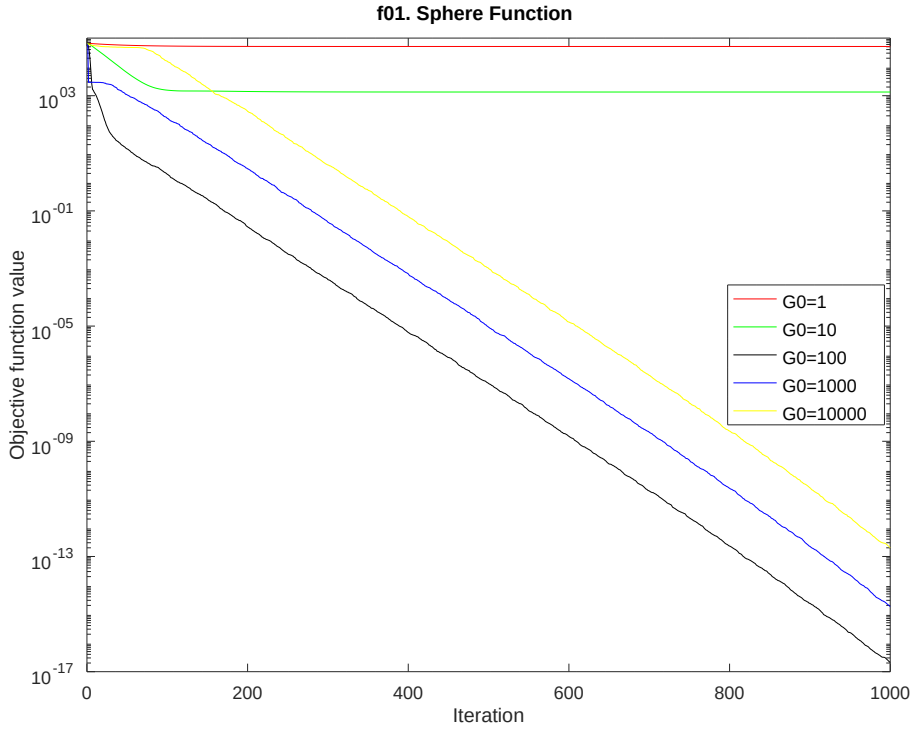

Figura 36 – A função de referência 1 variando a constante gravitacional inicial

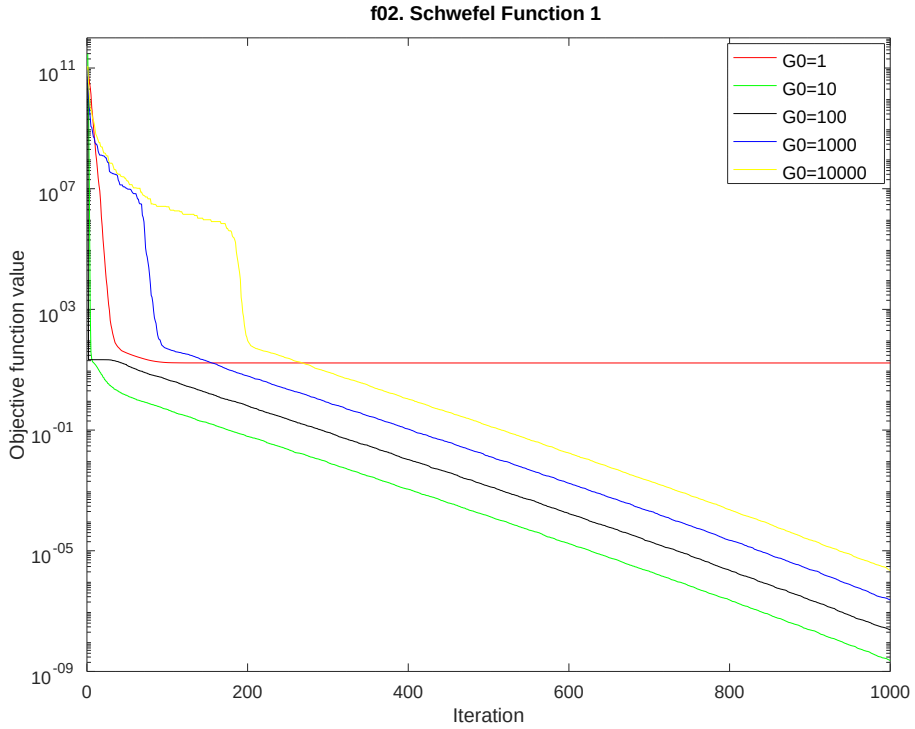

Figura 37 – A função de referência 2 variando a constante gravitacional inicial

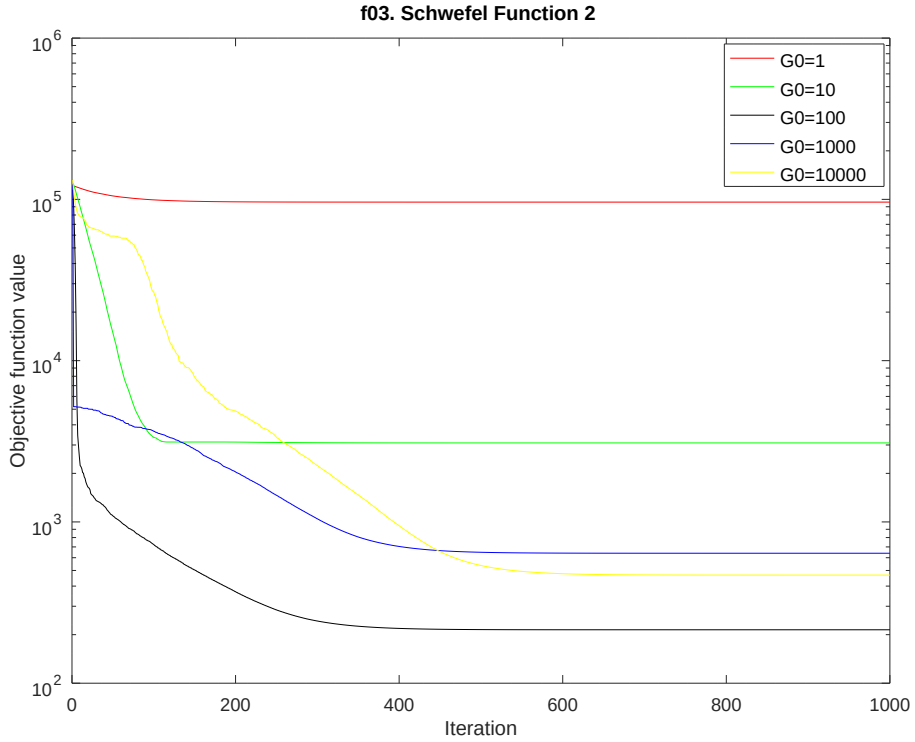

Figura 38 – A função de referência 3 variando a constante gravitacional inicial

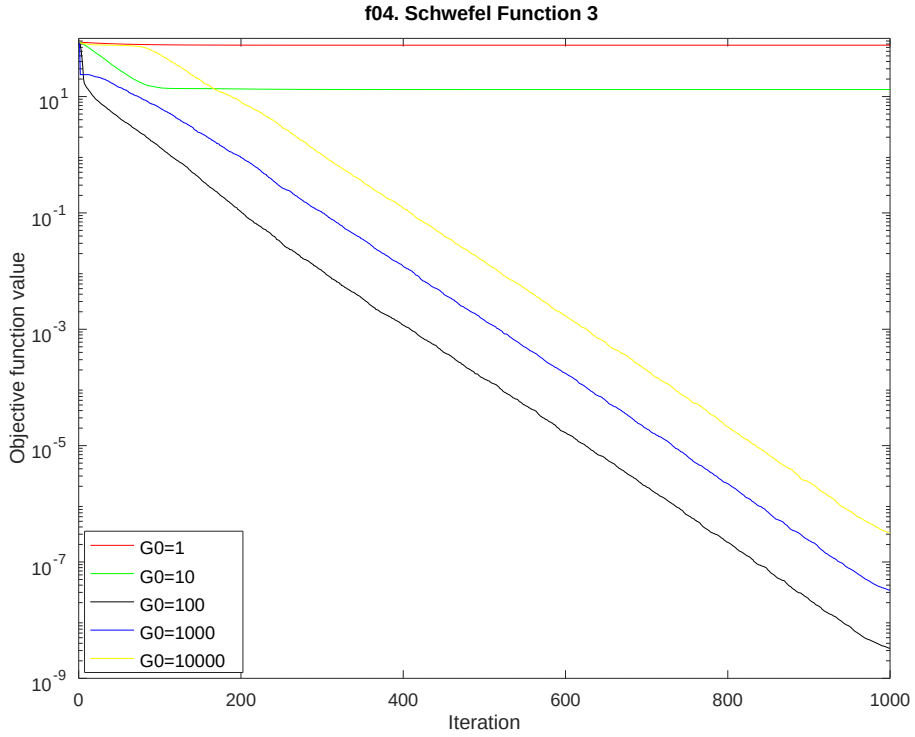

Figura 39 – A função de referência 4 variando a constante gravitacional inicial

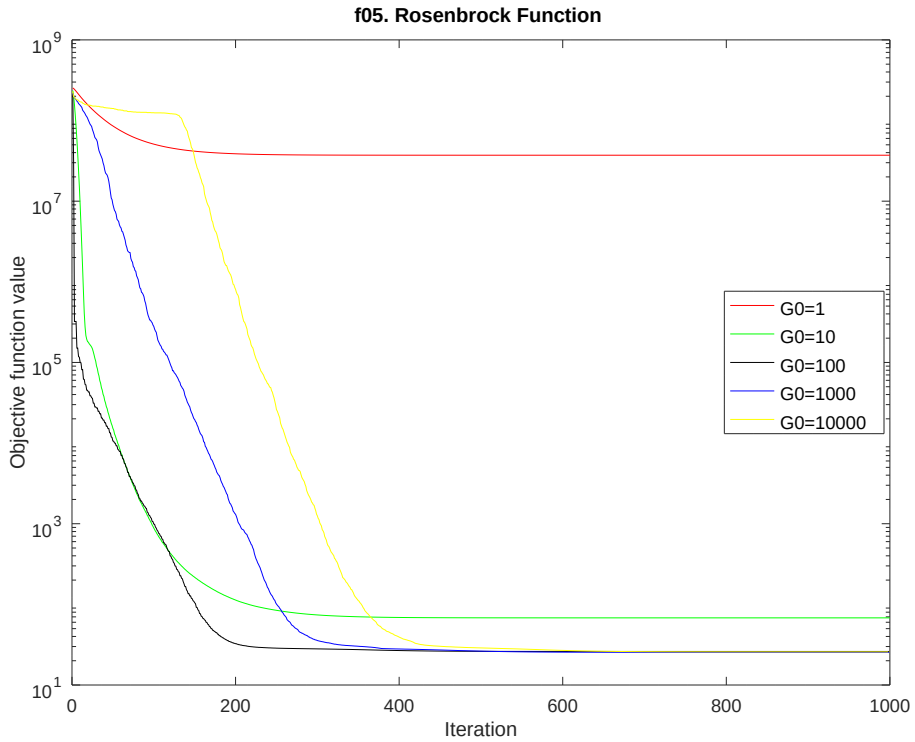

Figura 40 – A função de referência 5 variando a constante gravitacional inicial

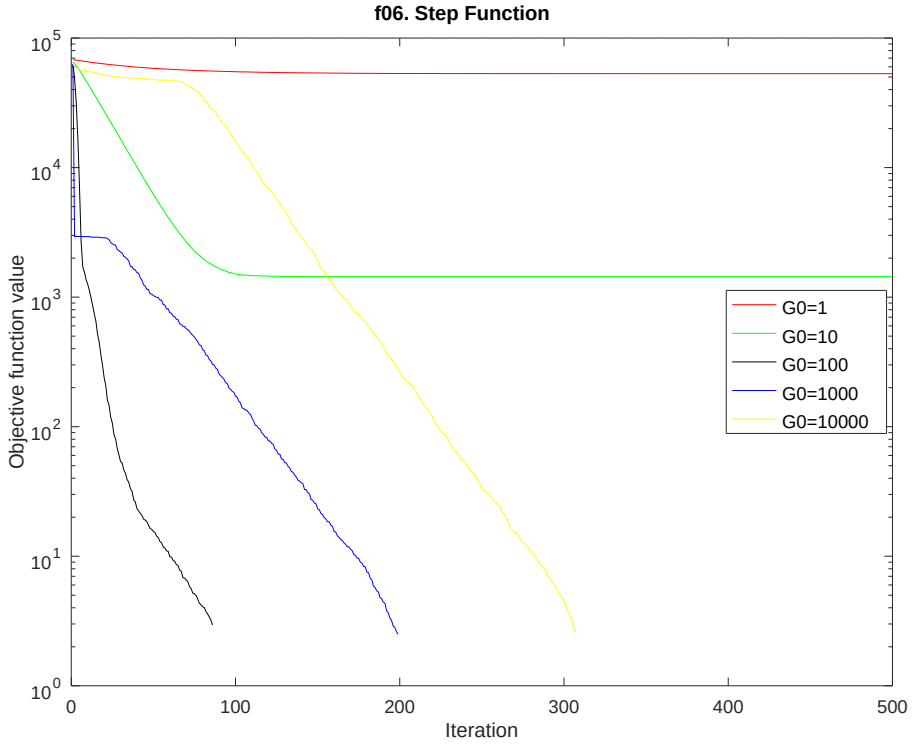

Figura 41 – A função de referência 6 variando a constante gravitacional inicial

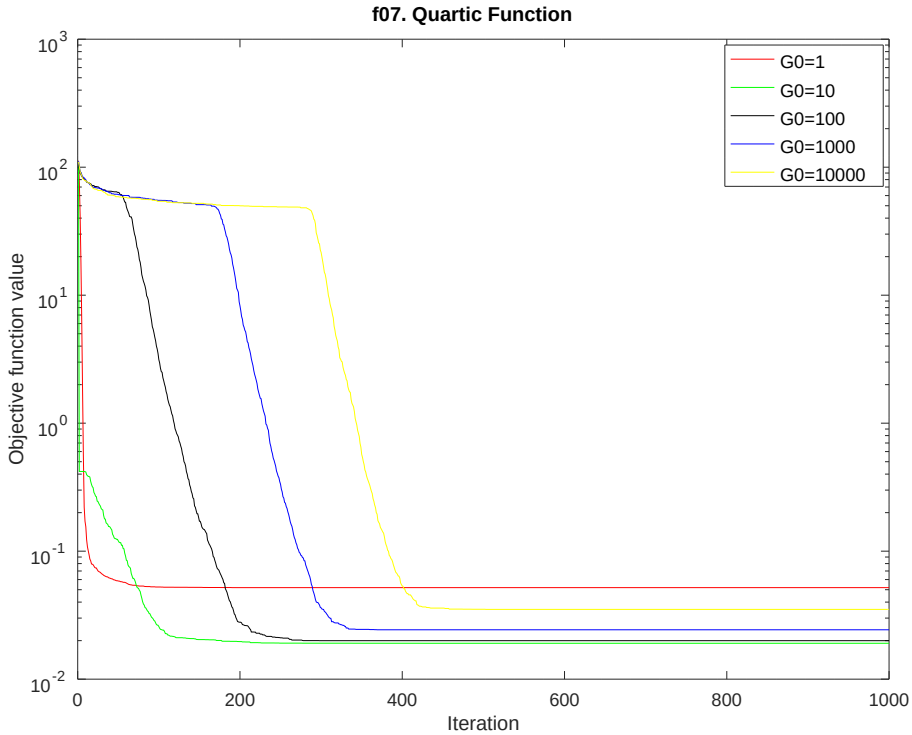

Figura 42 – A função de referência 7 variando a constante gravitacional inicial

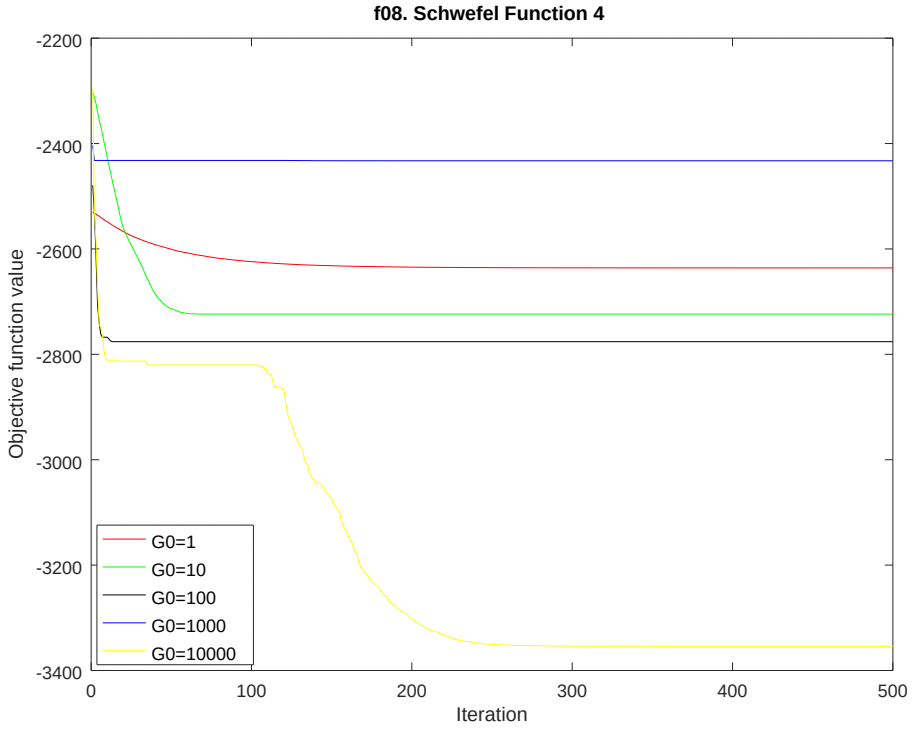

Figura 43 – A função de referência 8 variando a constante gravitacional inicial

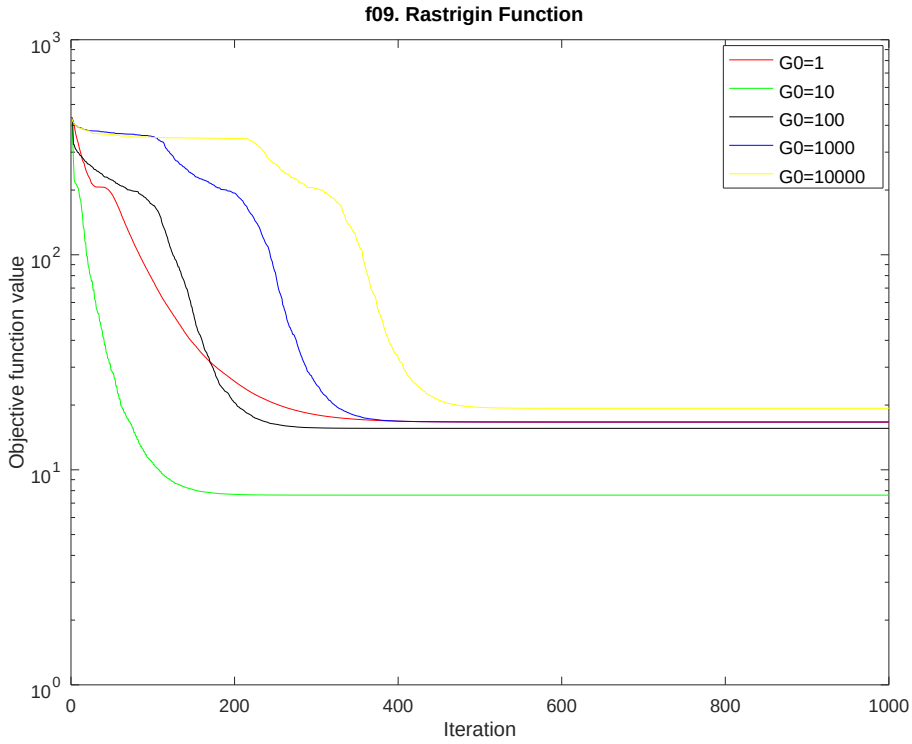

Figura 44 – A função de referência 9 variando a constante gravitacional inicial

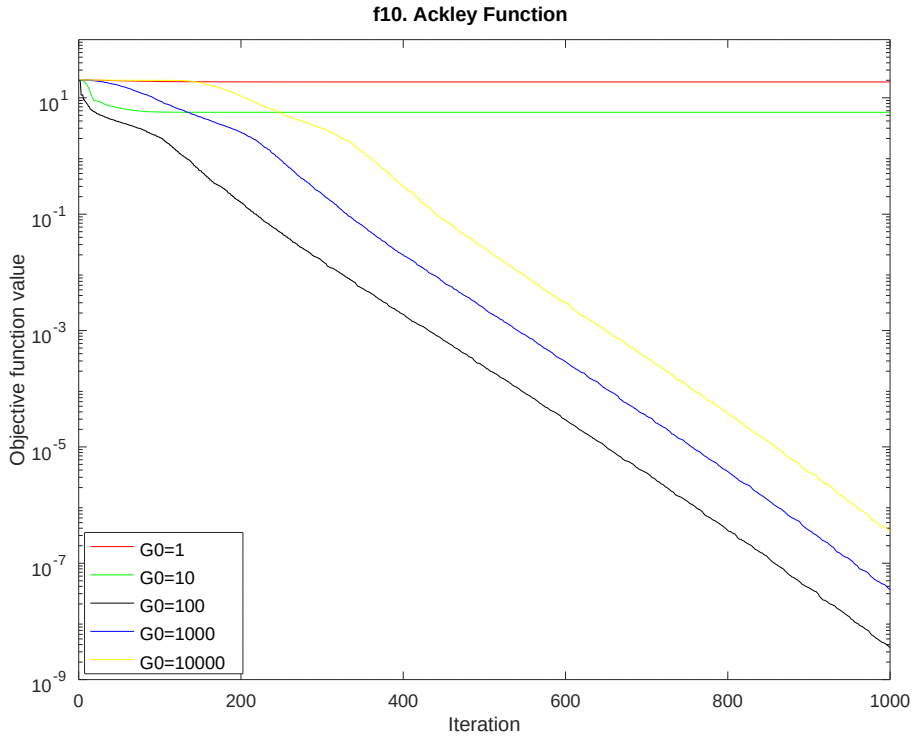

Figura 45 – A função de referência 10 variando a constante gravitacional inicial

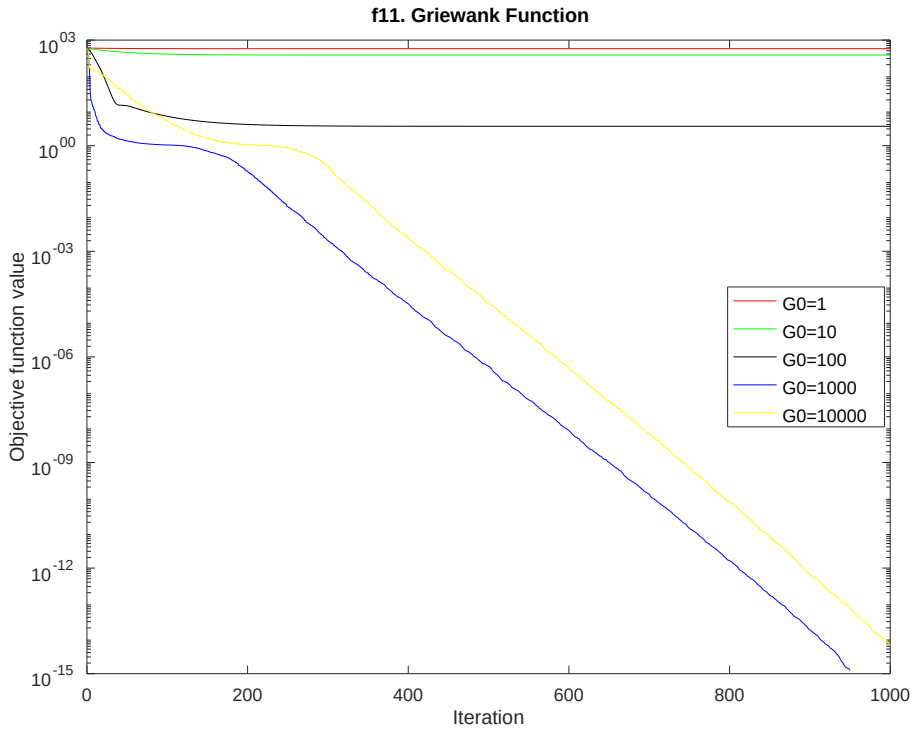

Figura 46 – A função de referência 11 variando a constante gravitacional inicial

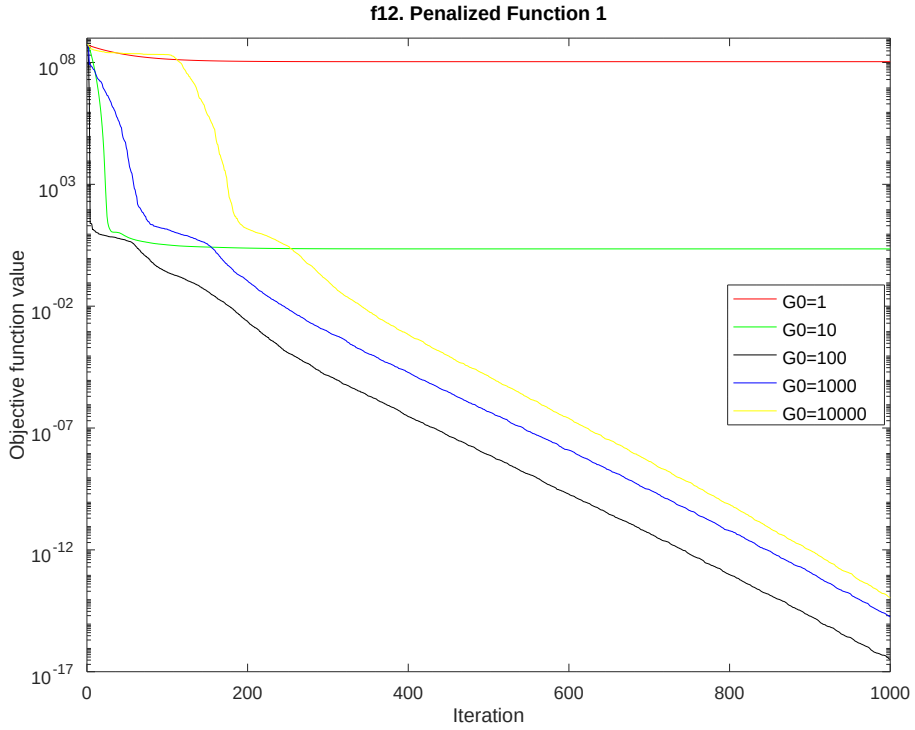

Figura 47 – A função de referência 12 variando a constante gravitacional inicial

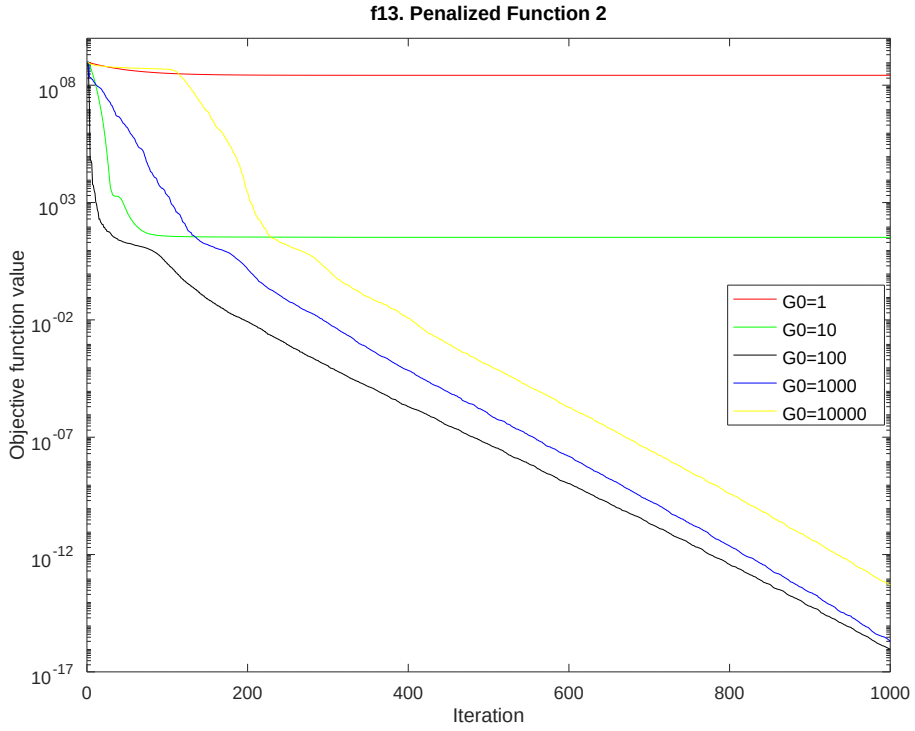

Figura 48 – A função de referência 13 variando a constante gravitacional inicial

APÊNDICE E – As funções de referência variando a constante β da heurística

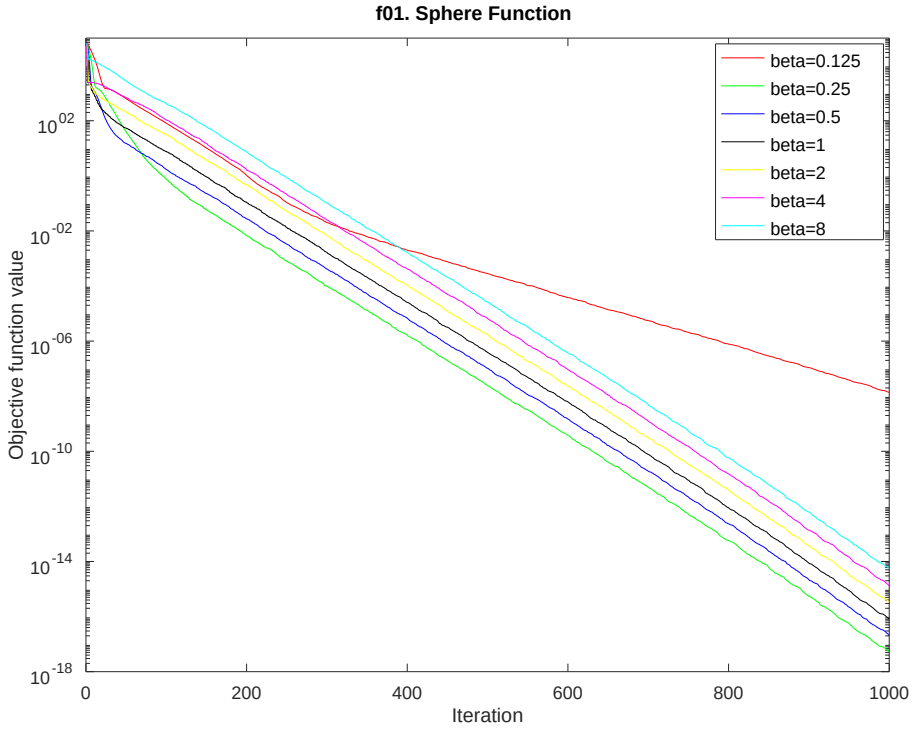

Figura 49 – A função de referência 1 variando a constante β da heurística

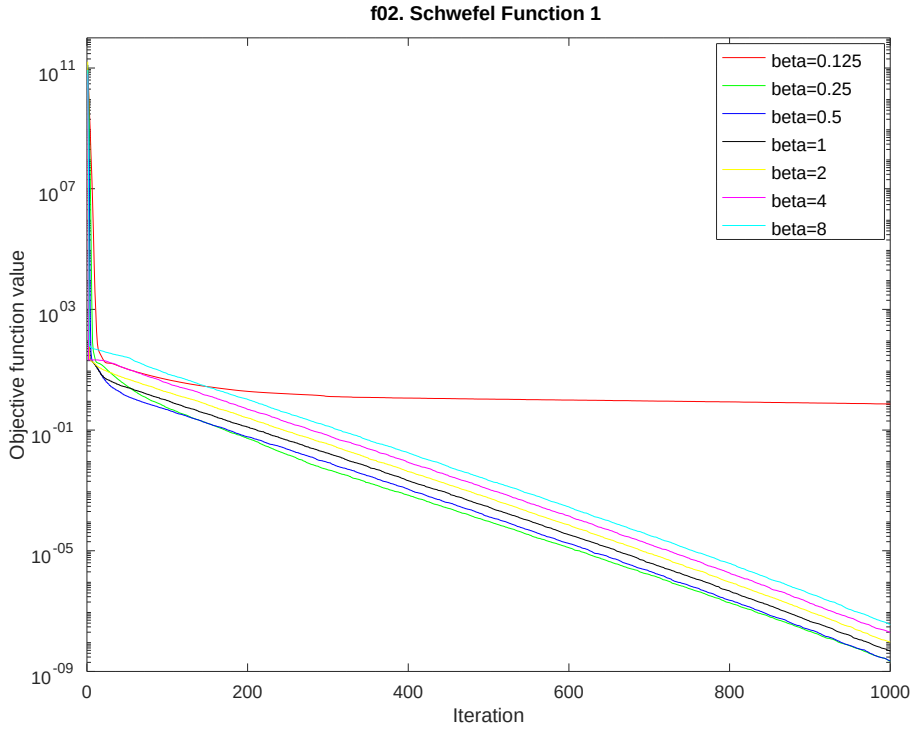

Figura 50 – A função de referência 2 variando a constante β da heurística

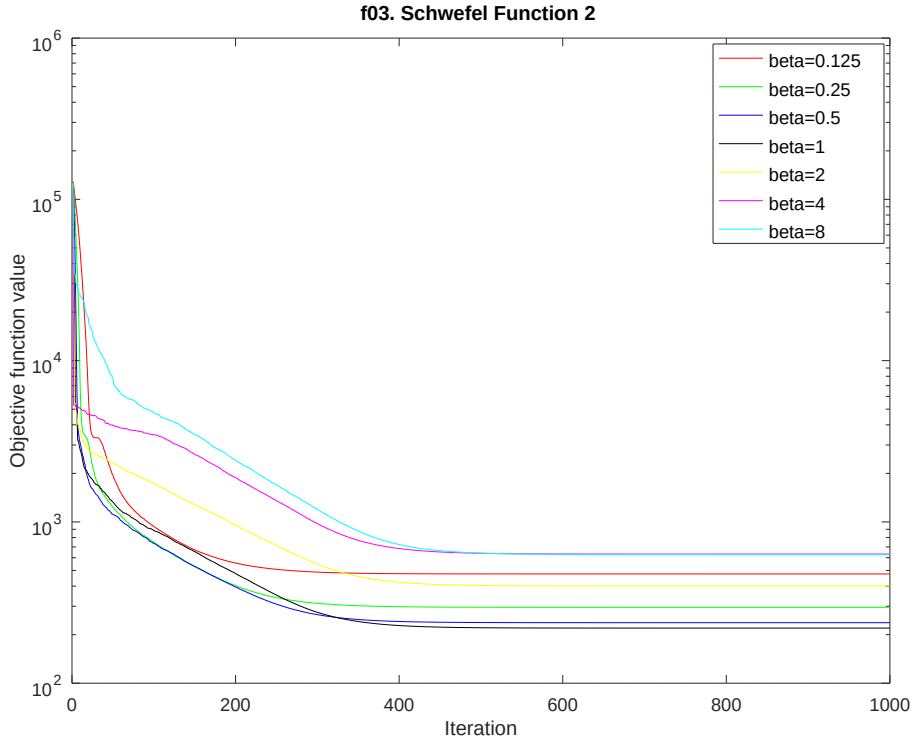

Figura 51 – A função de referência 3 variando a constante β da heurística

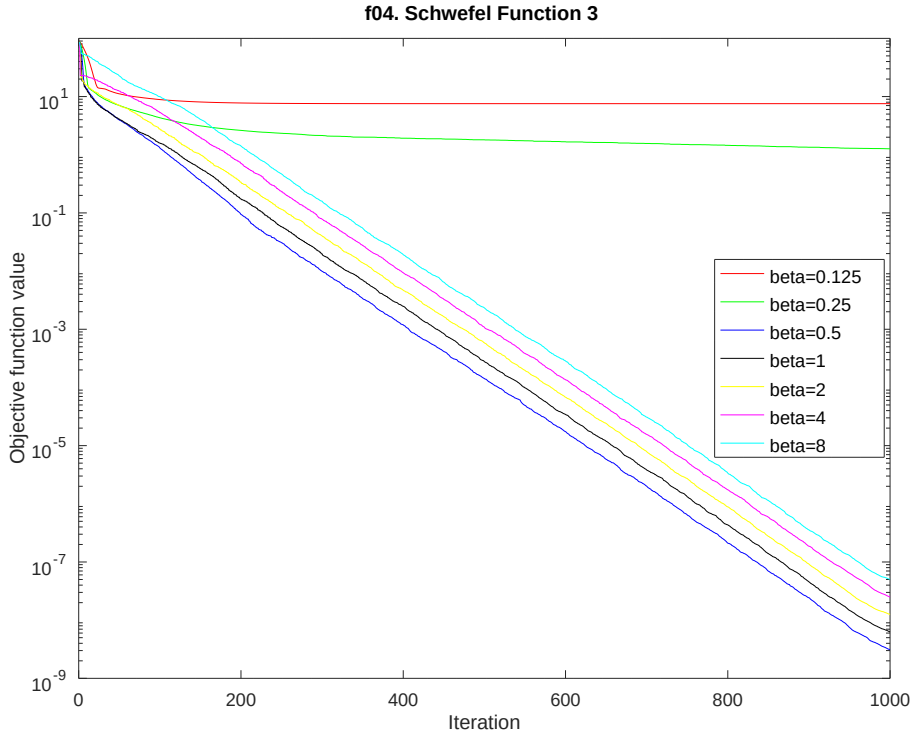

Figura 52 – A função de referência 4 variando a constante β da heurística

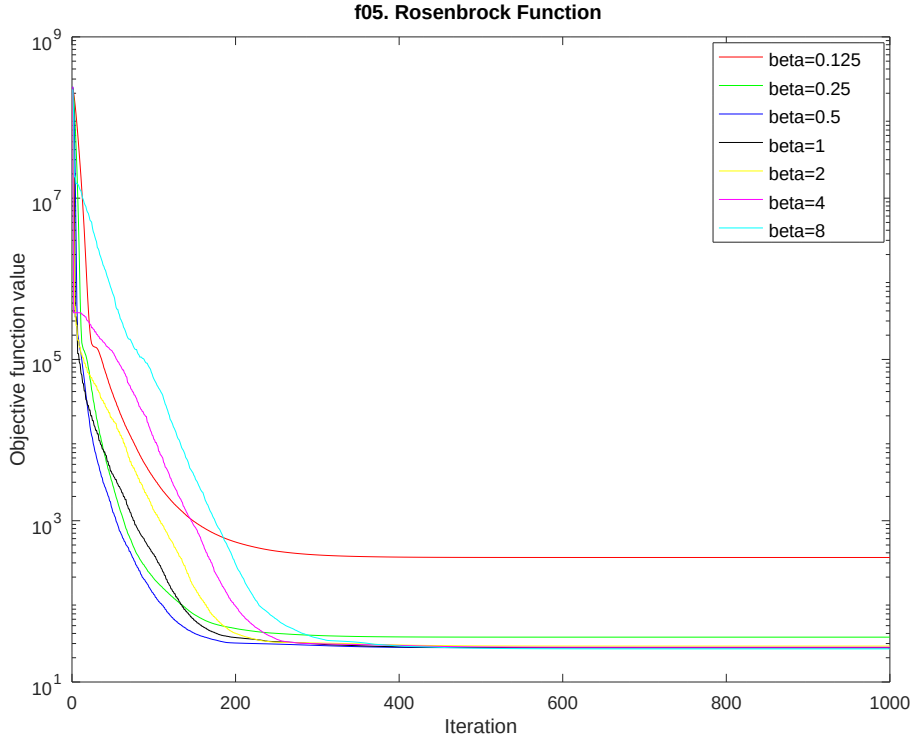

Figura 53 – A função de referência 5 variando a constante β da heurística

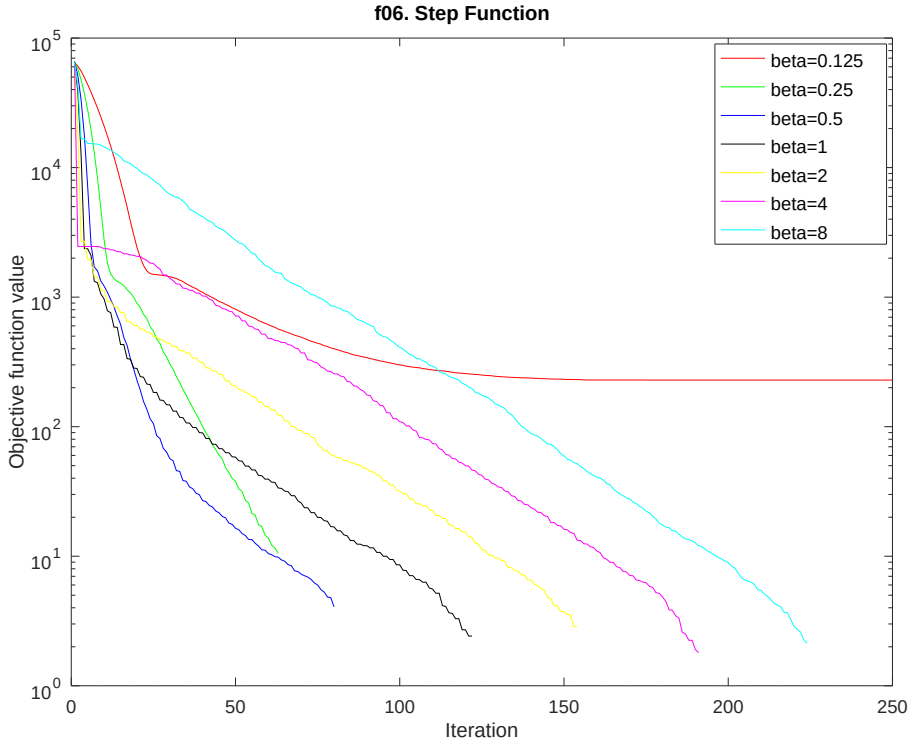

Figura 54 – A função de referência 6 variando a constante β da heurística

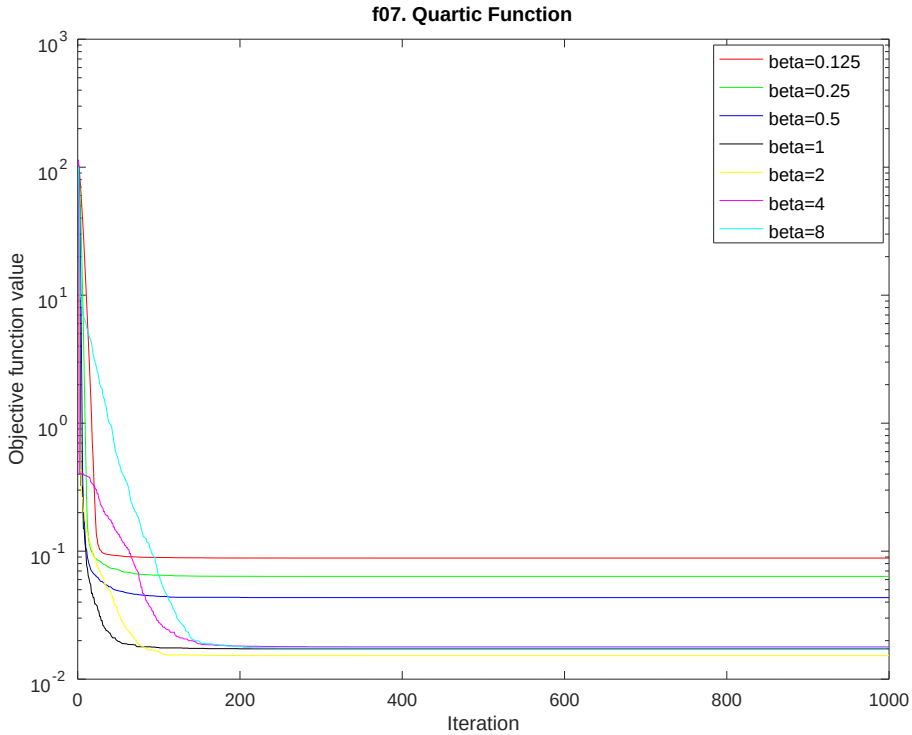

Figura 55 – A função de referência 7 variando a constante β da heurística

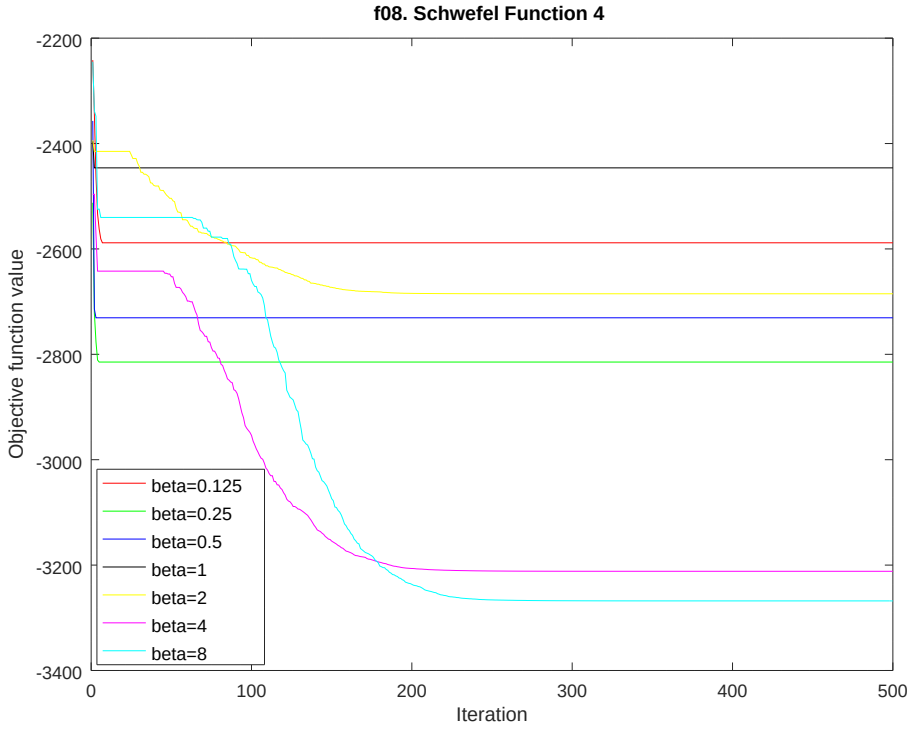

Figura 56 – A função de referência 8 variando a constante β da heurística

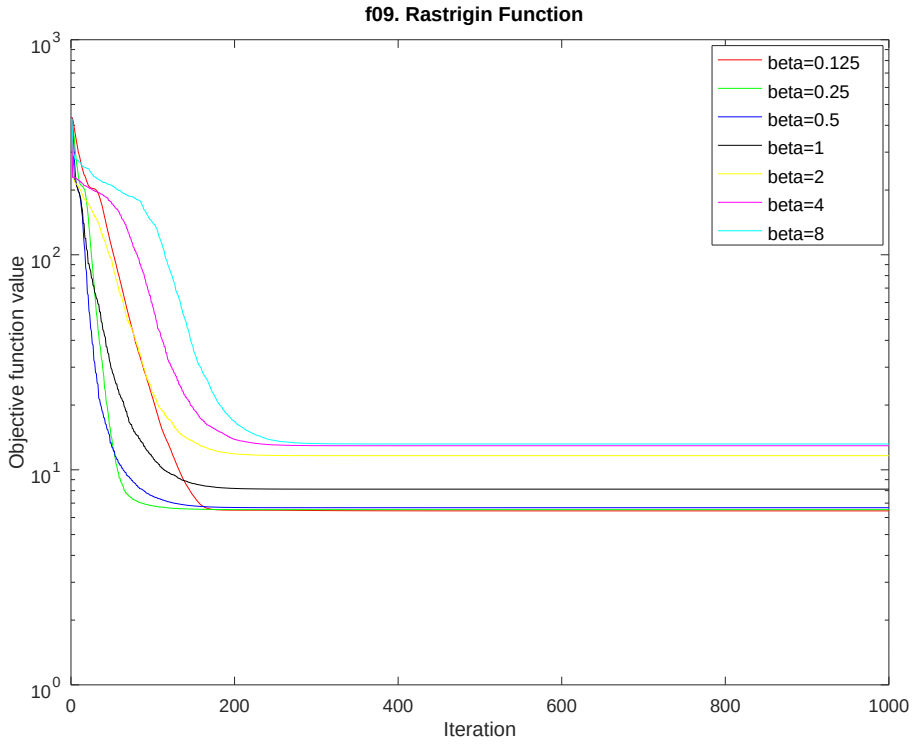

Figura 57 – A função de referência 9 variando a constante β da heurística

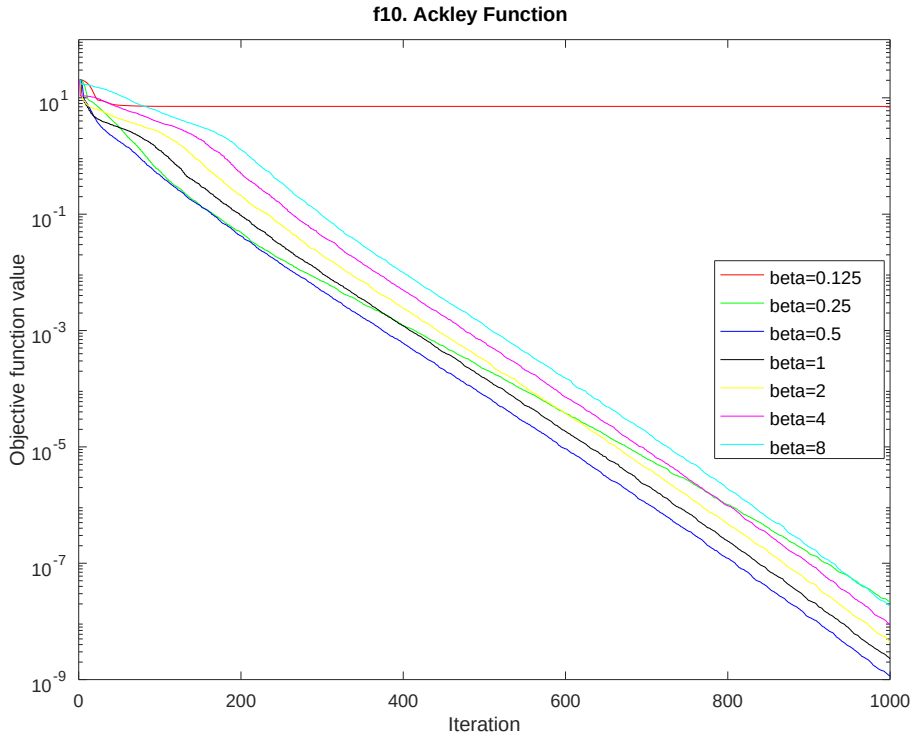

Figura 58 – A função de referência 10 variando a constante β da heurística

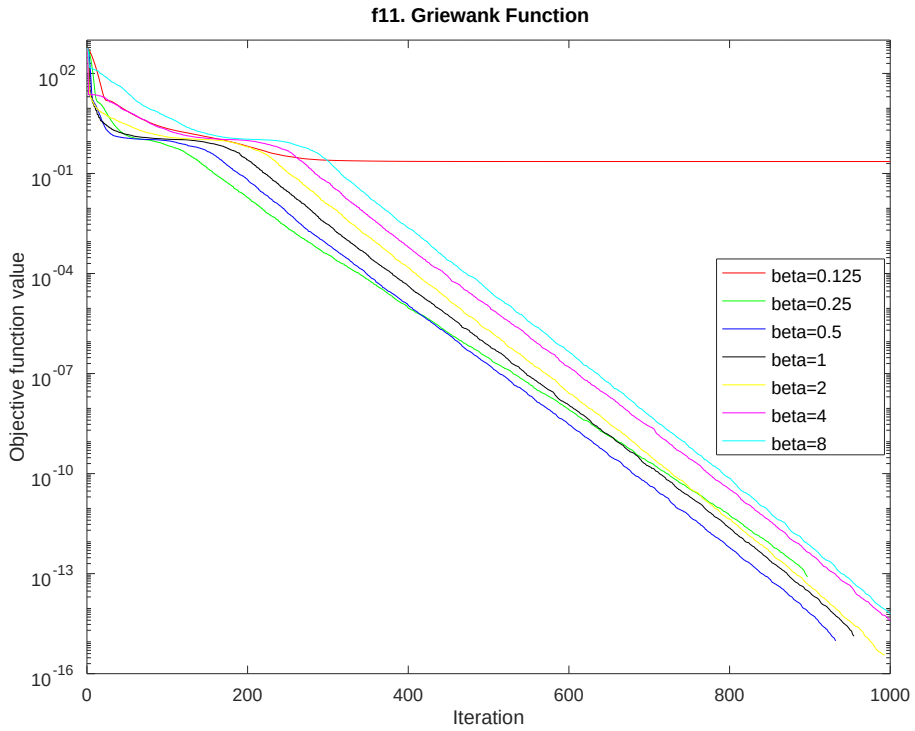

Figura 59 – A função de referência 11 variando a constante β da heurística

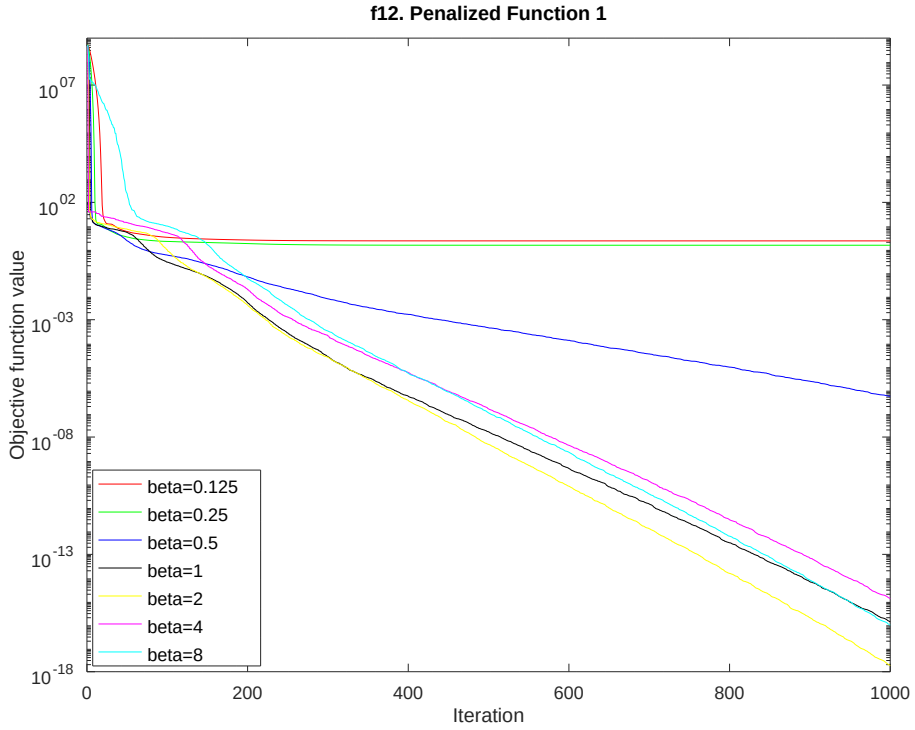

Figura 60 – A função de referência 12 variando a constante β da heurística

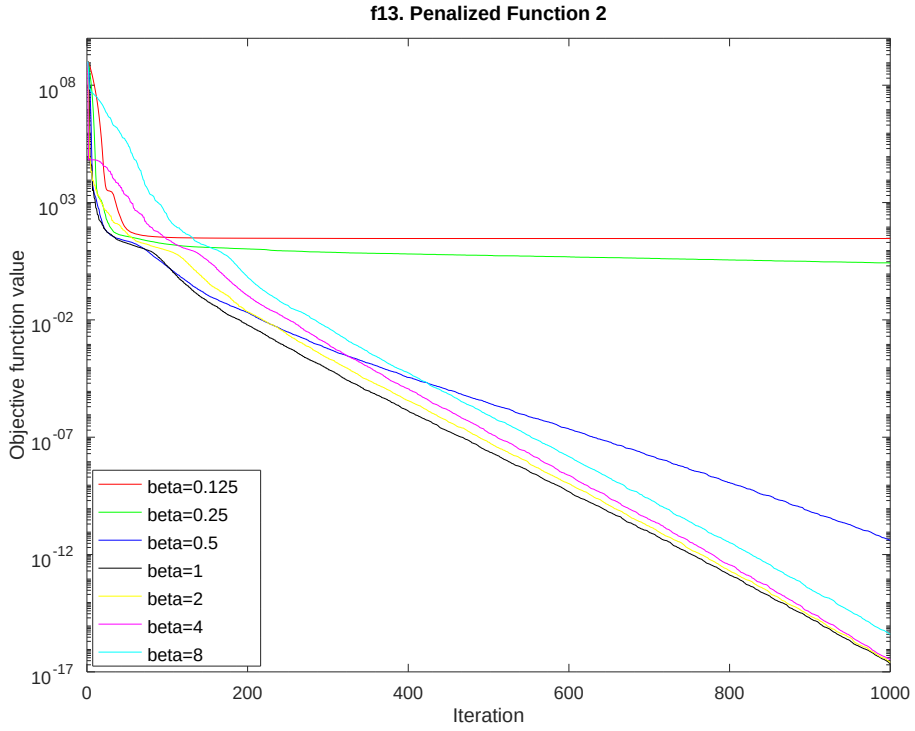

Figura 61 – A função de referência 13 variando a constante β da heurística

APÊNDICE F – As funções de referência avaliadas em um espaço de busca pequeno

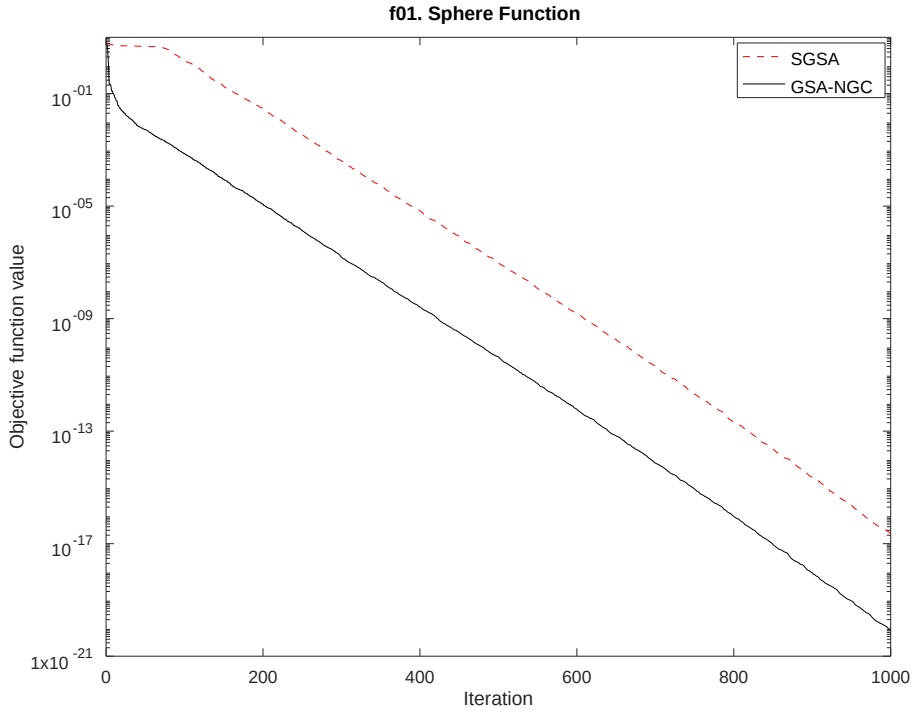

Figura 62 – A função de referência 1 avaliada em um espaço de busca pequeno

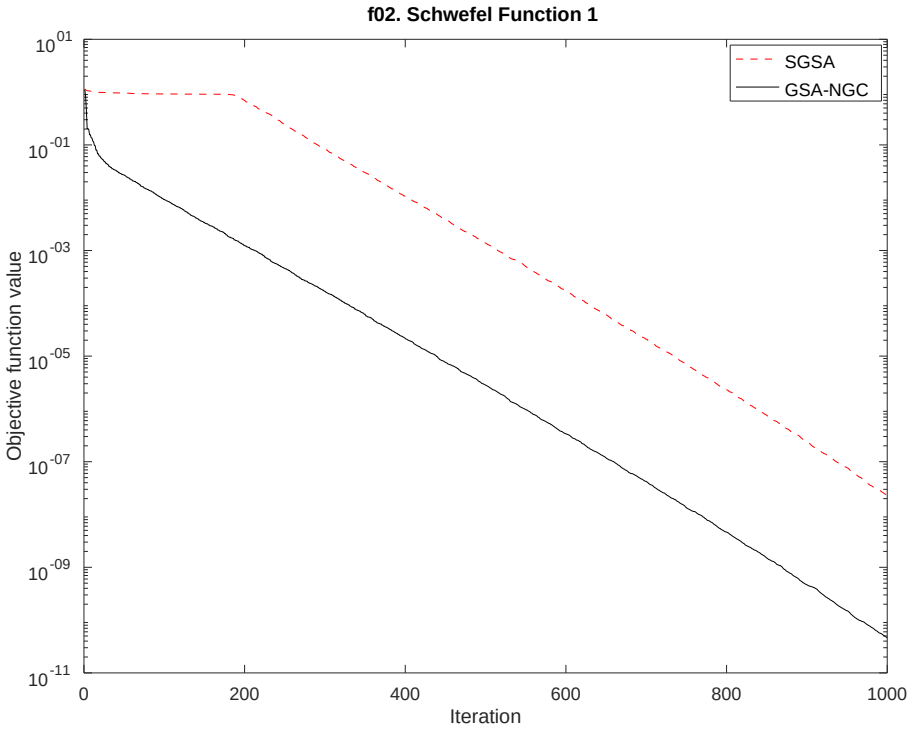

Figura 63 – A função de referência 2 avaliada em um espaço de busca pequeno

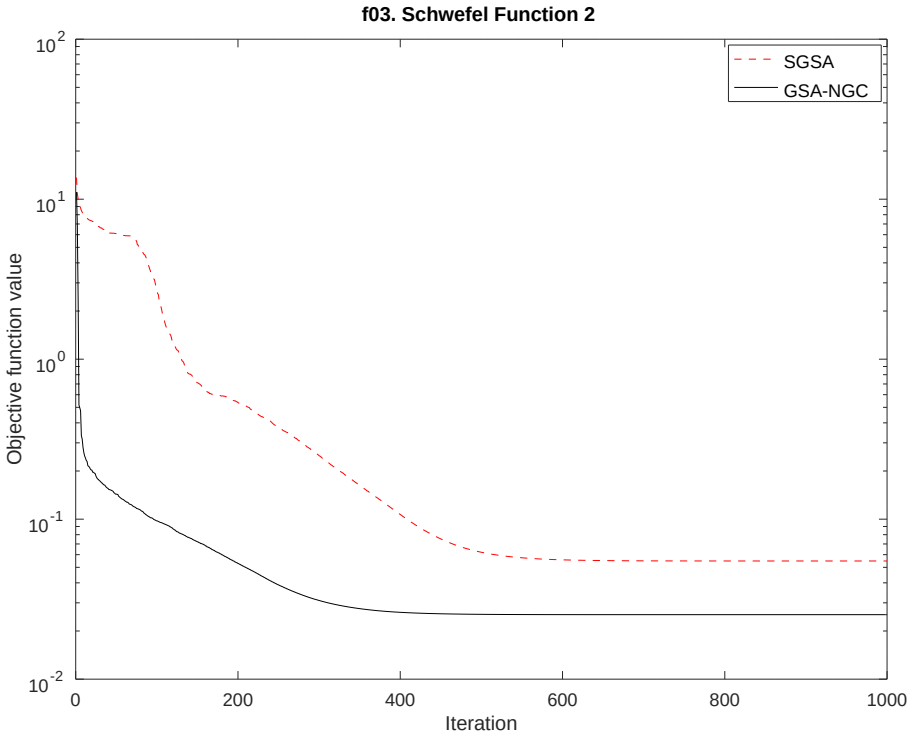

Figura 64 – A função de referência 3 avaliada em um espaço de busca pequeno

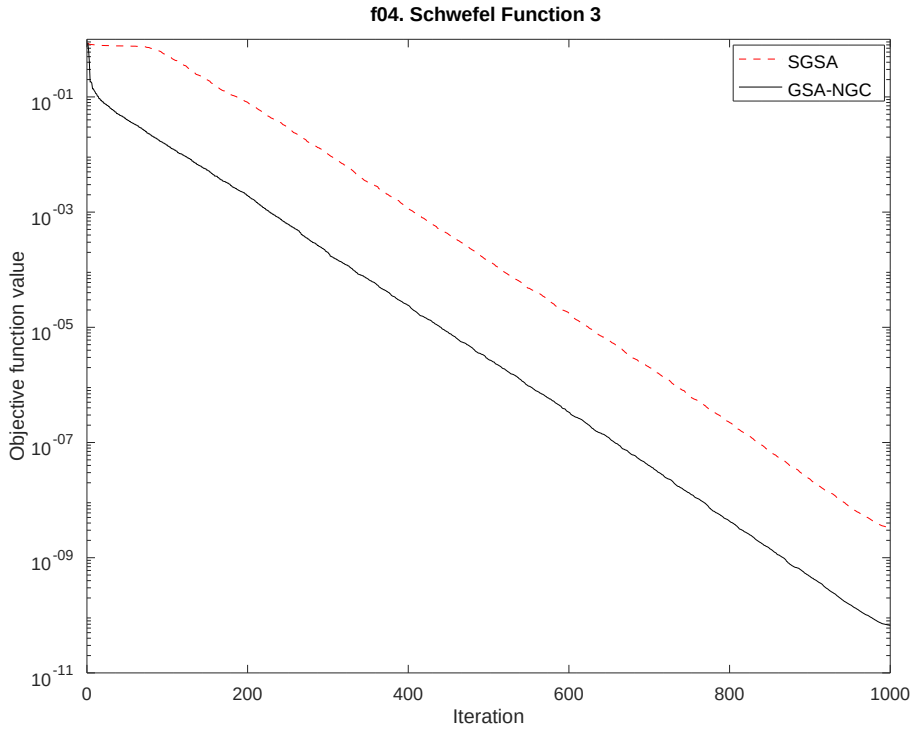

Figura 65 – A função de referência 4 avaliada em um espaço de busca pequeno

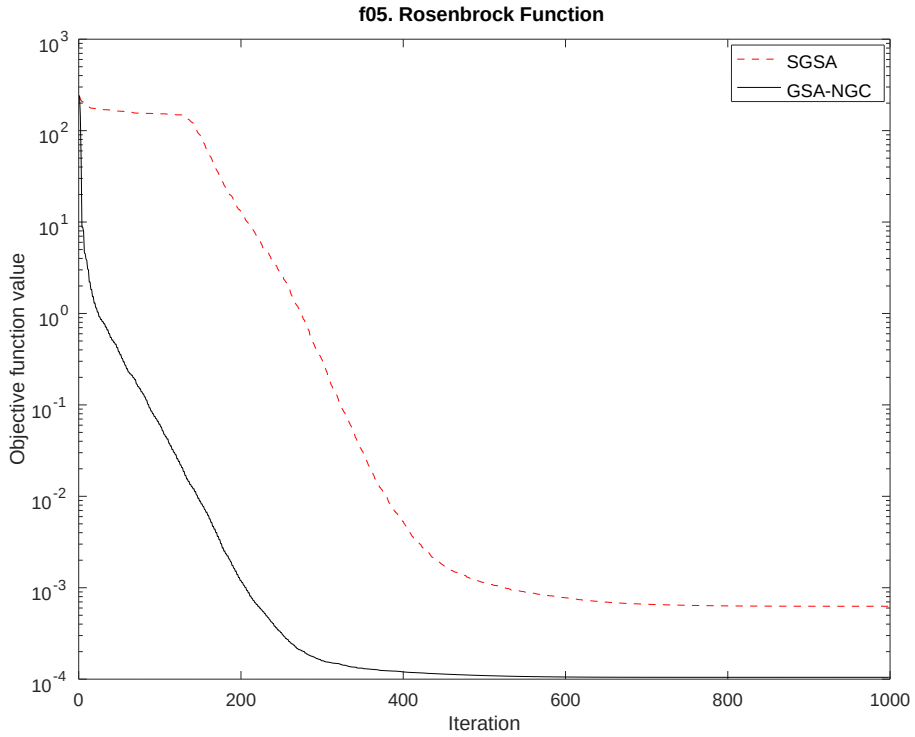

Figura 66 – A função de referência 5 avaliada em um espaço de busca pequeno

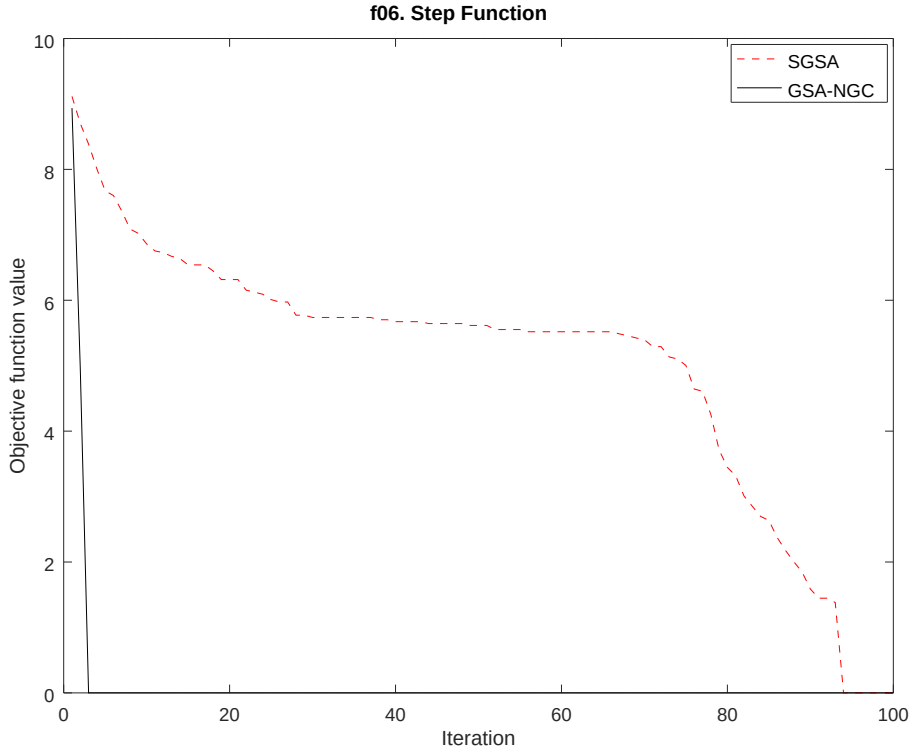

Figura 67 – A função de referência 6 avaliada em um espaço de busca pequeno

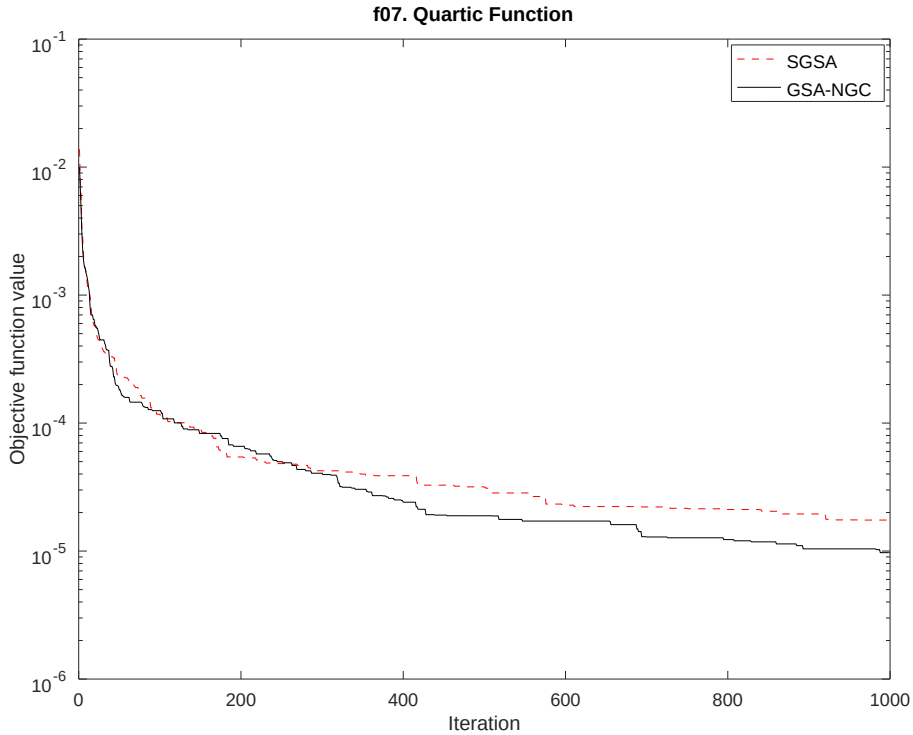

Figura 68 – A função de referência 7 avaliada em um espaço de busca pequeno

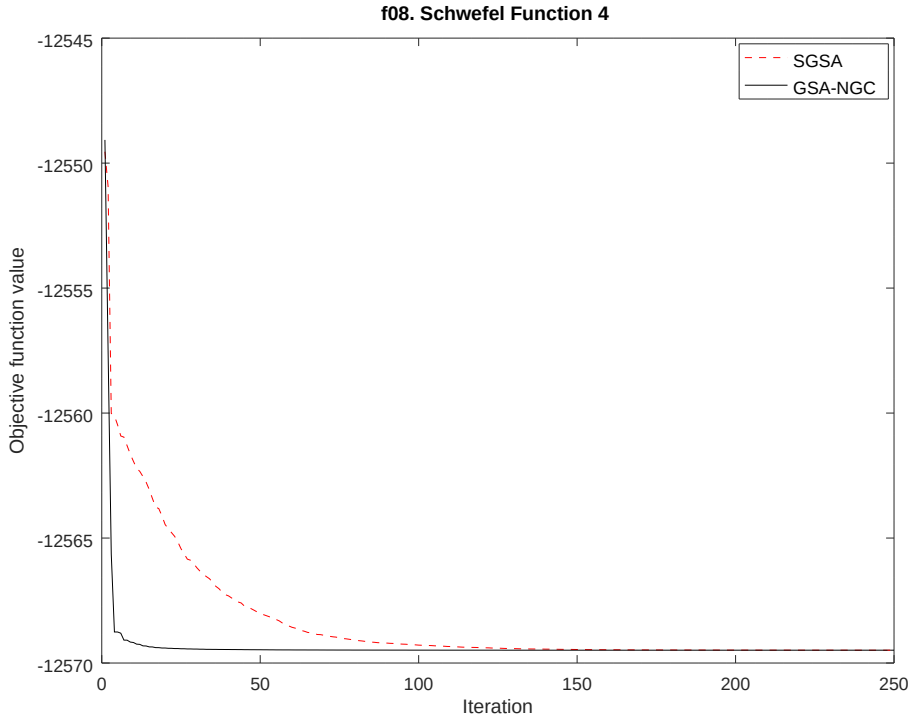

Figura 69 – A função de referência 8 avaliada em um espaço de busca pequeno

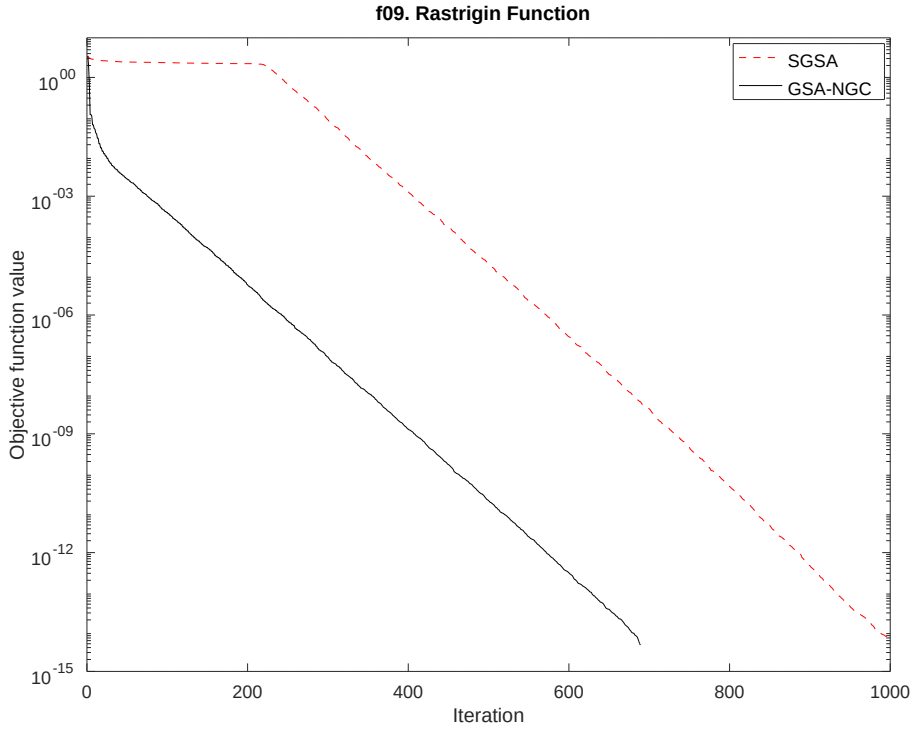

Figura 70 – A função de referência 9 avaliada em um espaço de busca pequeno

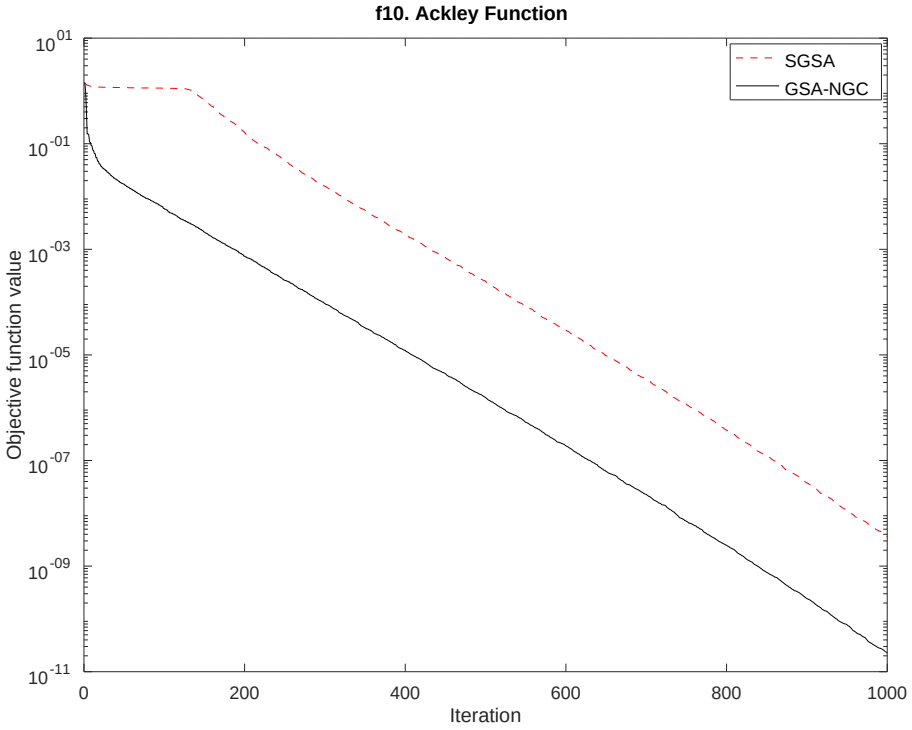

Figura 71 – A função de referência 10 avaliada em um espaço de busca pequeno

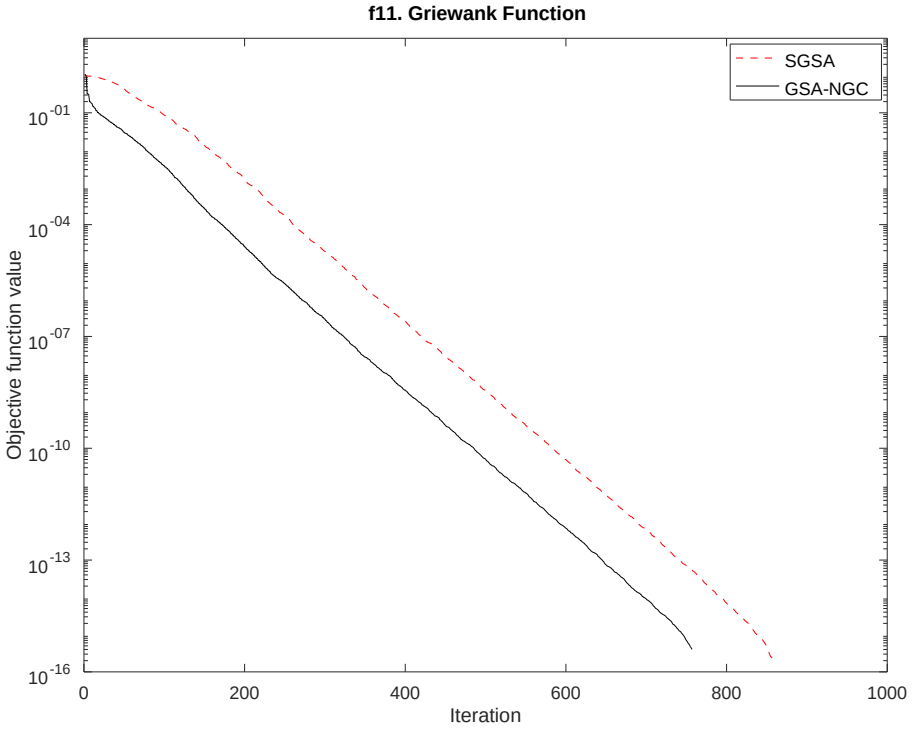

Figura 72 – A função de referência 11 avaliada em um espaço de busca pequeno

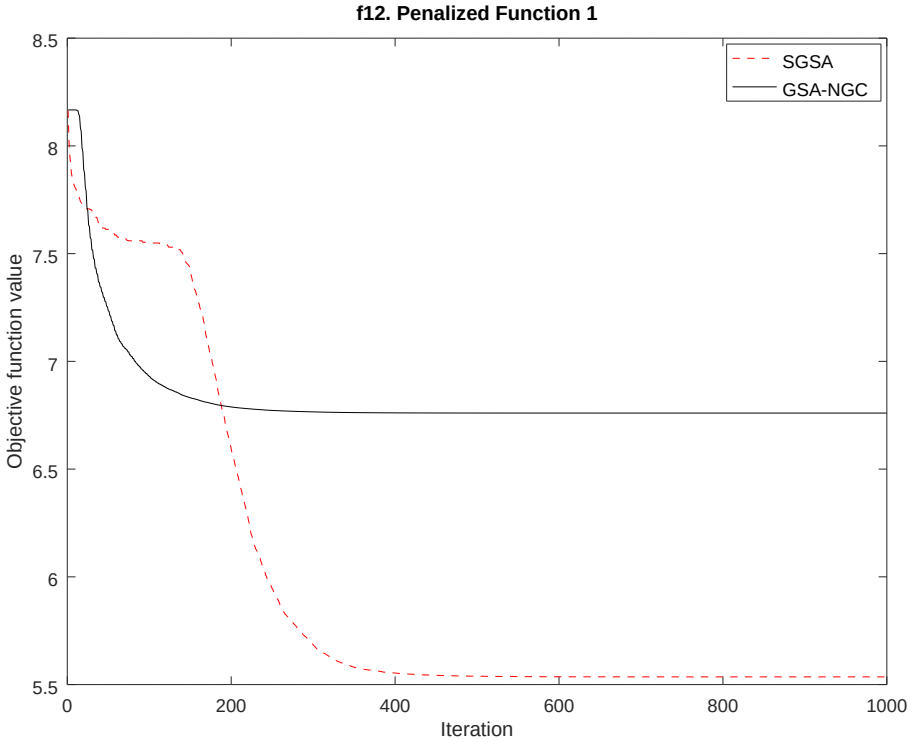

Figura 73 – A função de referência 12 avaliada em um espaço de busca pequeno

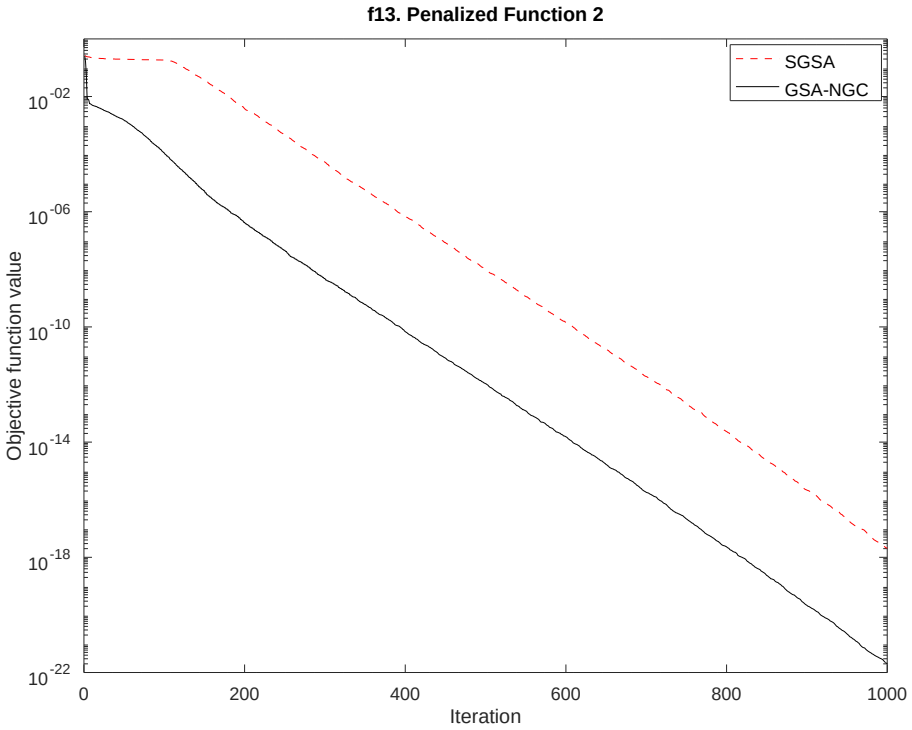

Figura 74 – A função de referência 13 avaliada em um espaço de busca pequeno

APÊNDICE G – As funções de referência avaliadas em um espaço de busca grande

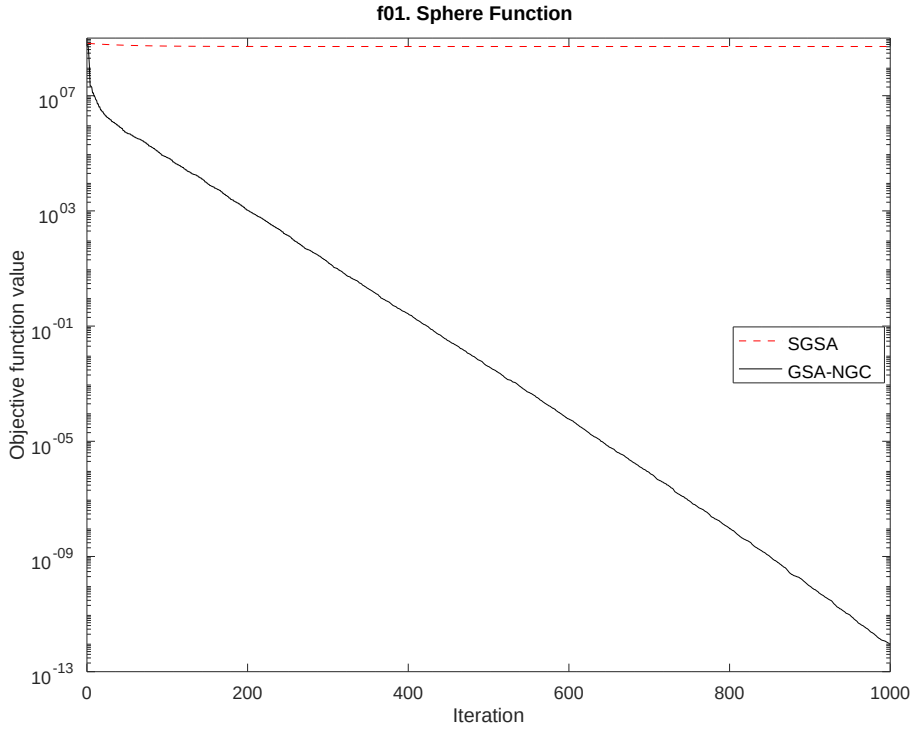

Figura 75 – A função de referência 1 avaliada em um espaço de busca grande

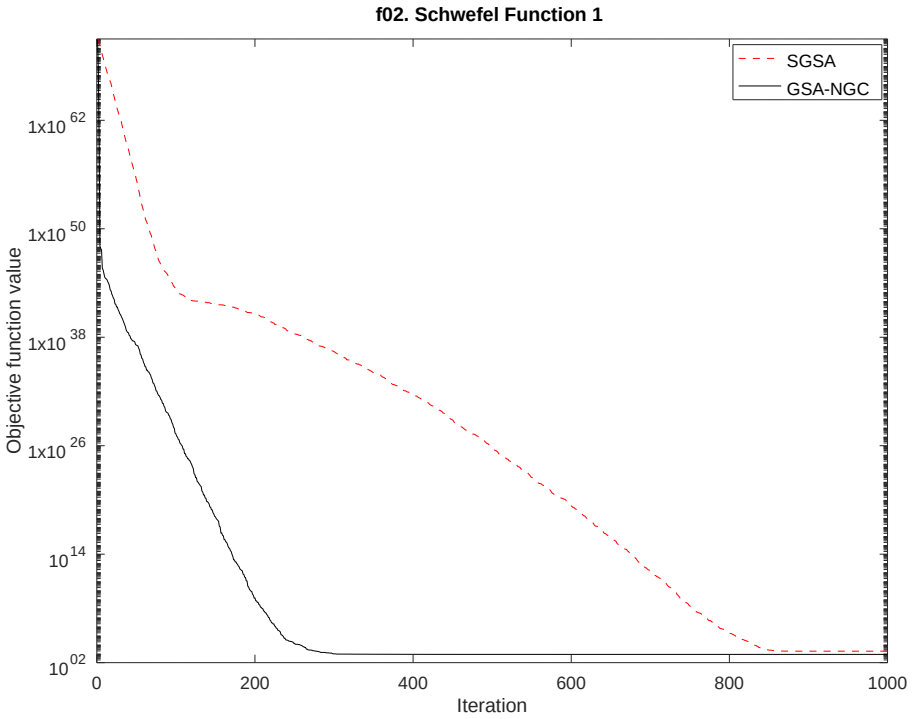

Figura 76 – A função de referência 2 avaliada em um espaço de busca grande

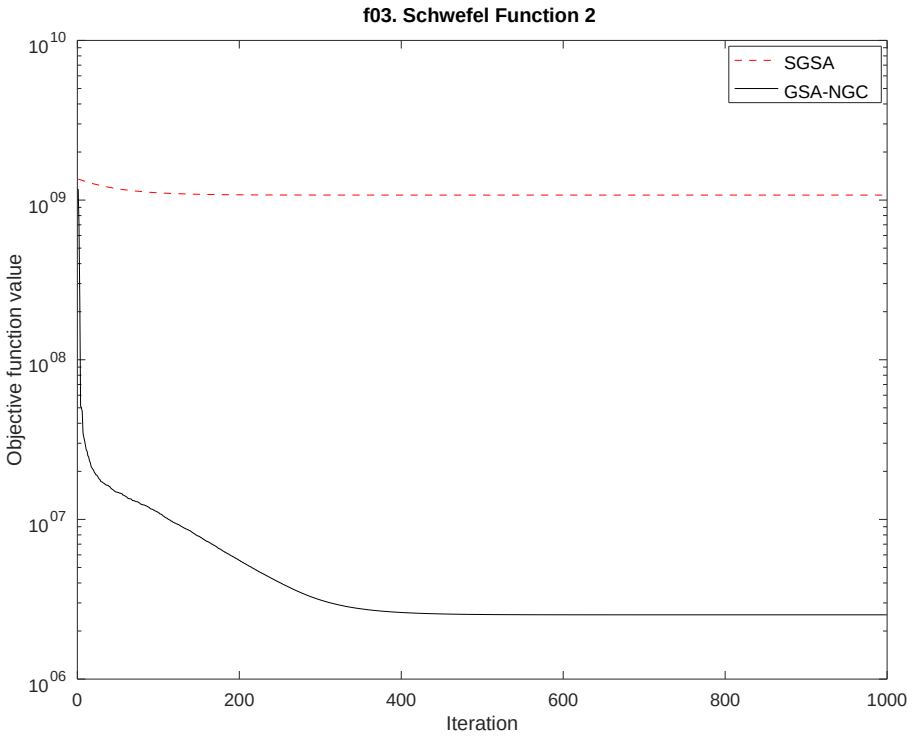

Figura 77 – A função de referência 3 avaliada em um espaço de busca grande

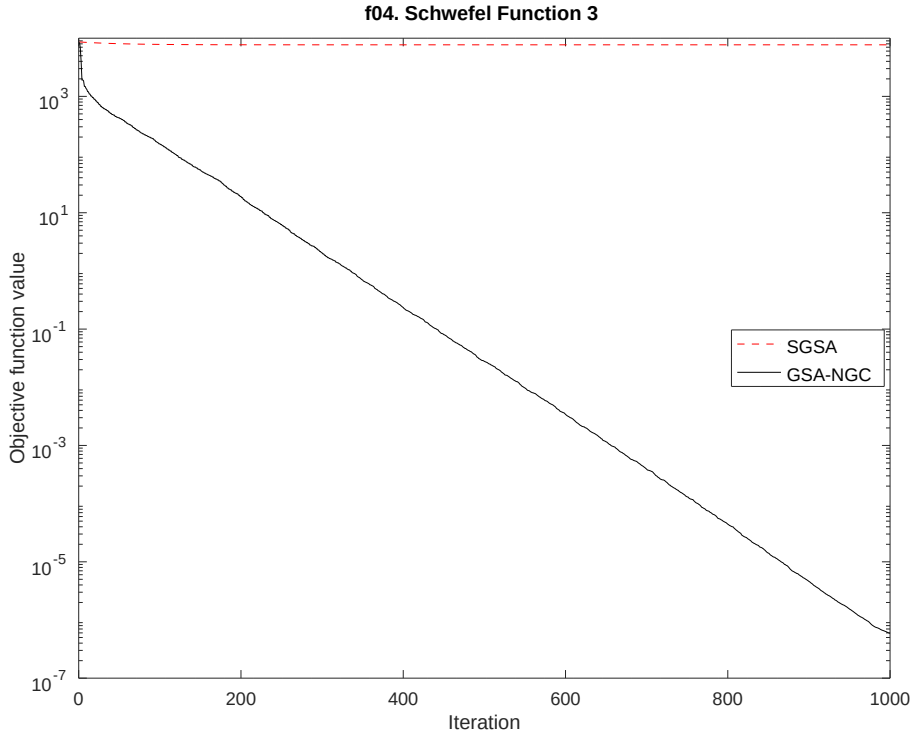

Figura 78 – A função de referência 4 avaliada em um espaço de busca grande

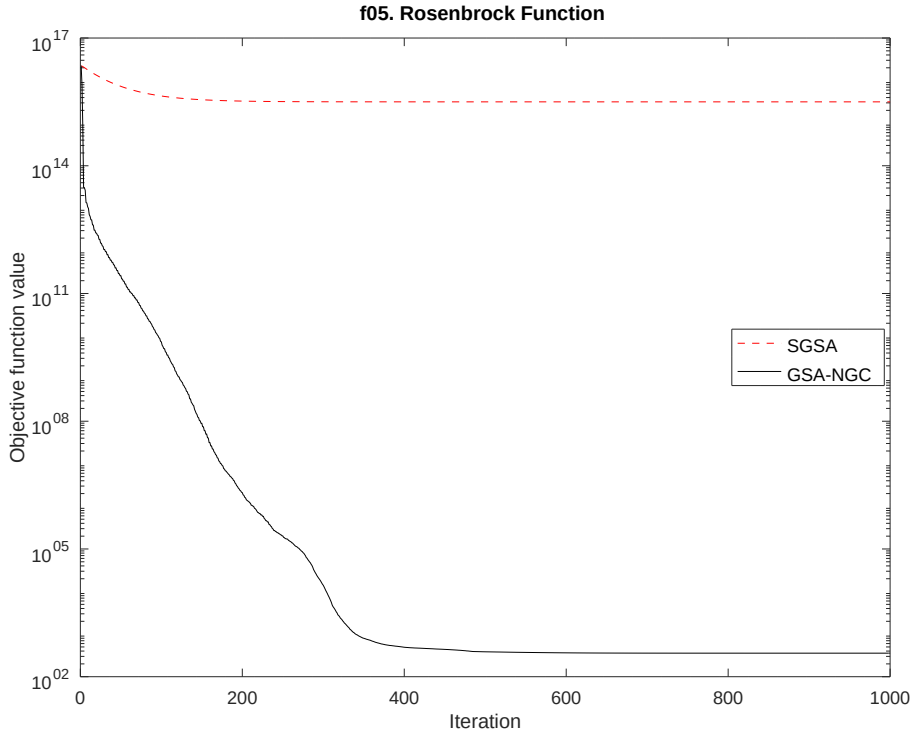

Figura 79 – A função de referência 5 avaliada em um espaço de busca grande

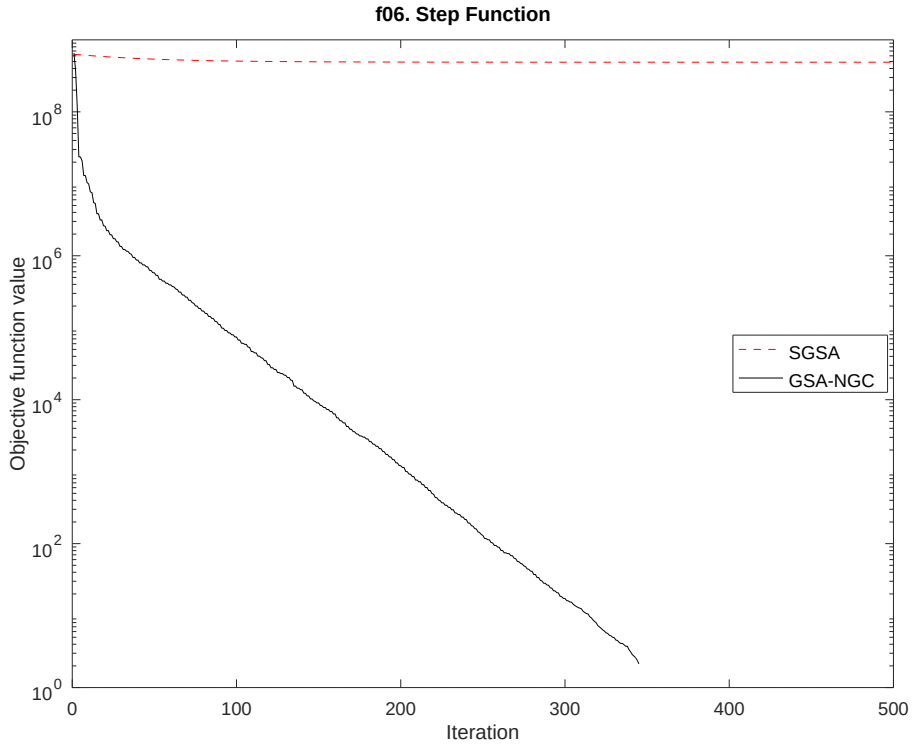

Figura 80 – A função de referência 6 avaliada em um espaço de busca grande

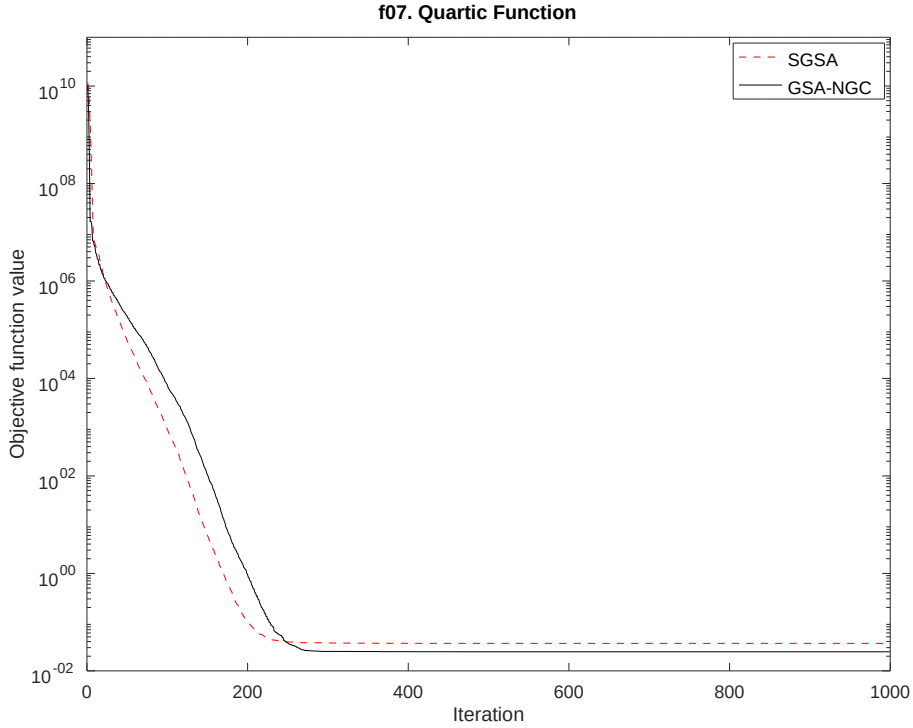

Figura 81 – A função de referência 7 avaliada em um espaço de busca grande

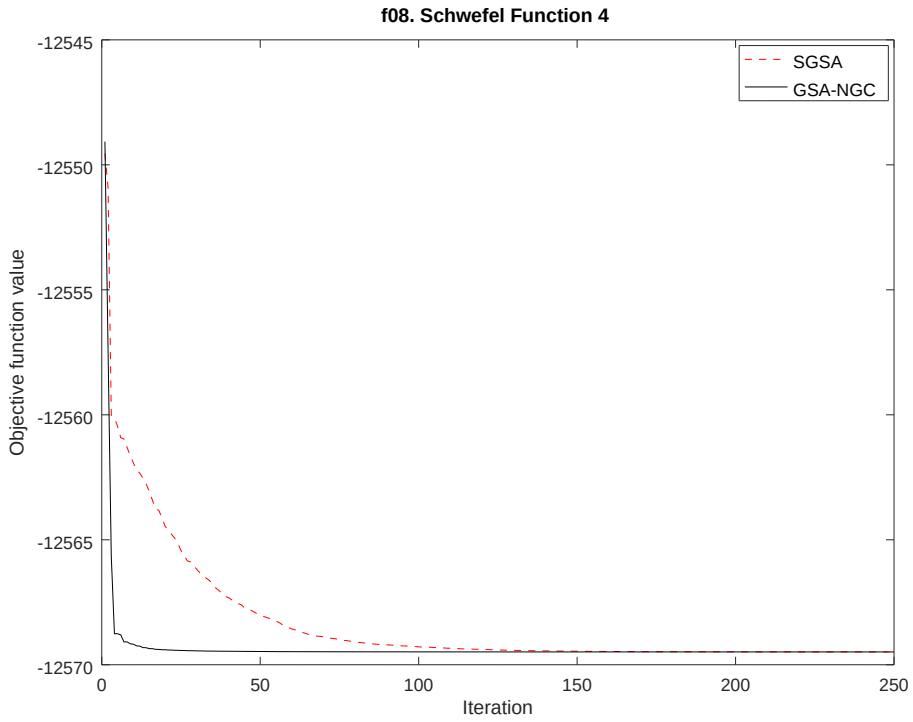

Figura 82 – A função de referência 8 avaliada em um espaço de busca grande

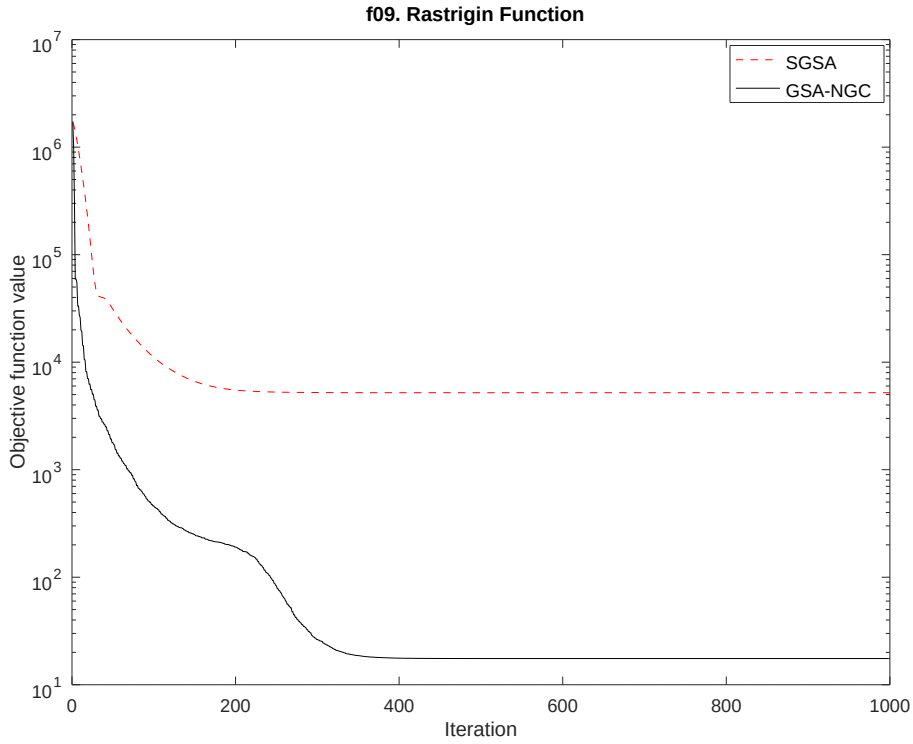

Figura 83 – A função de referência 9 avaliada em um espaço de busca grande

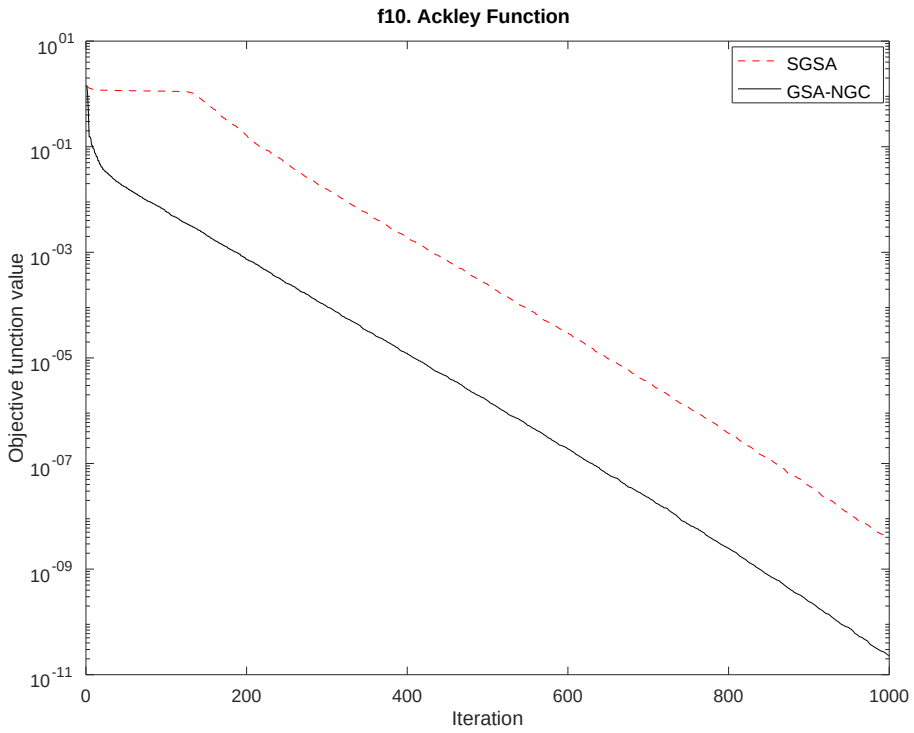

Figura 84 – A função de referência 10 avaliada em um espaço de busca grande

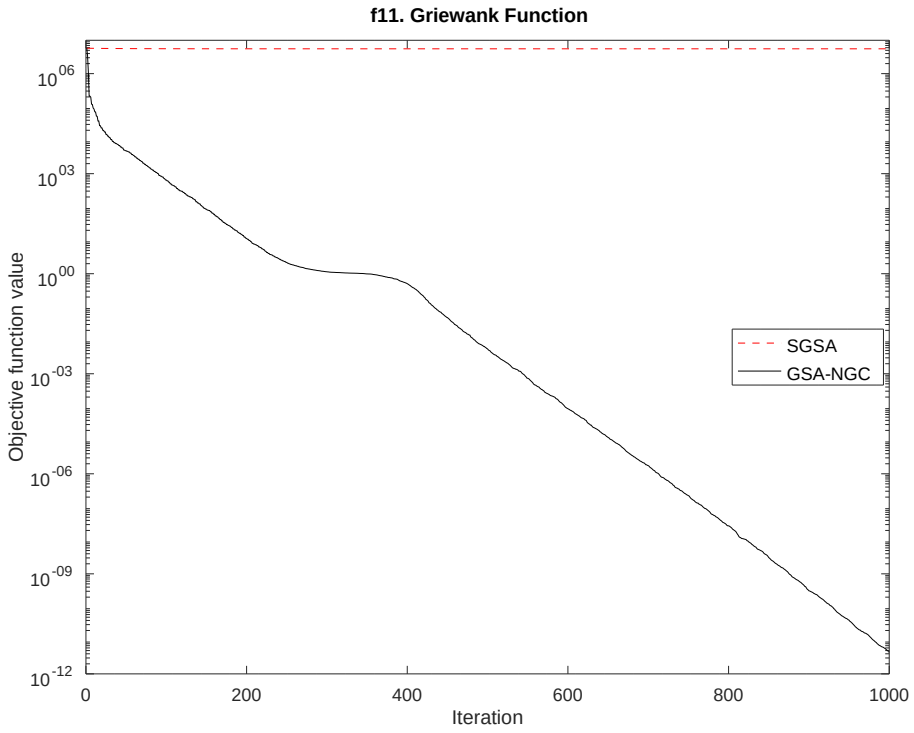

Figura 85 – A função de referência 11 avaliada em um espaço de busca grande

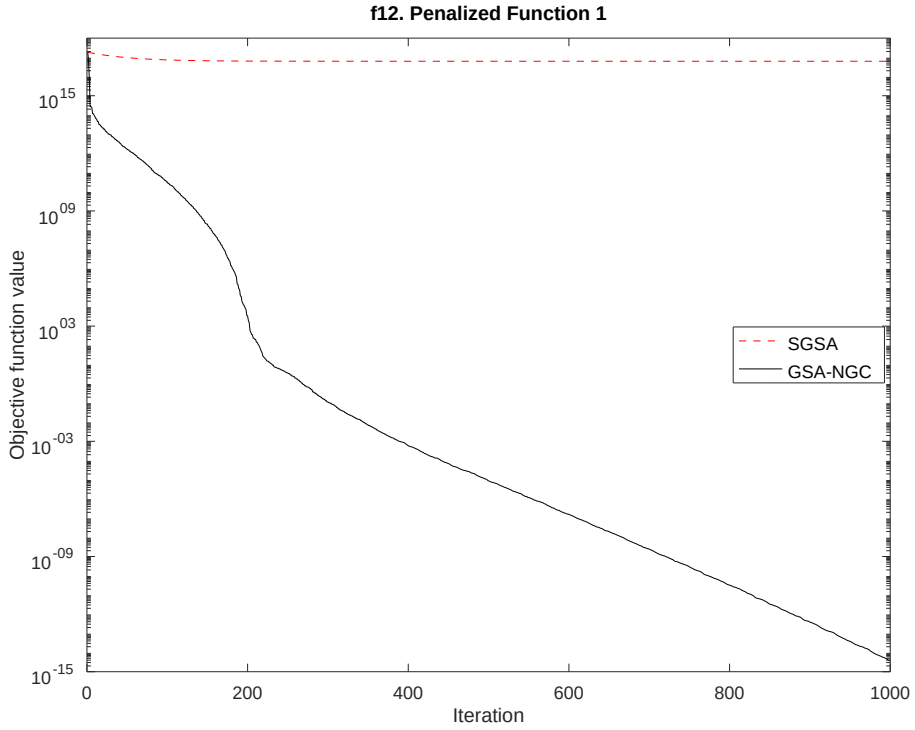

Figura 86 – A função de referência 12 avaliada em um espaço de busca grande

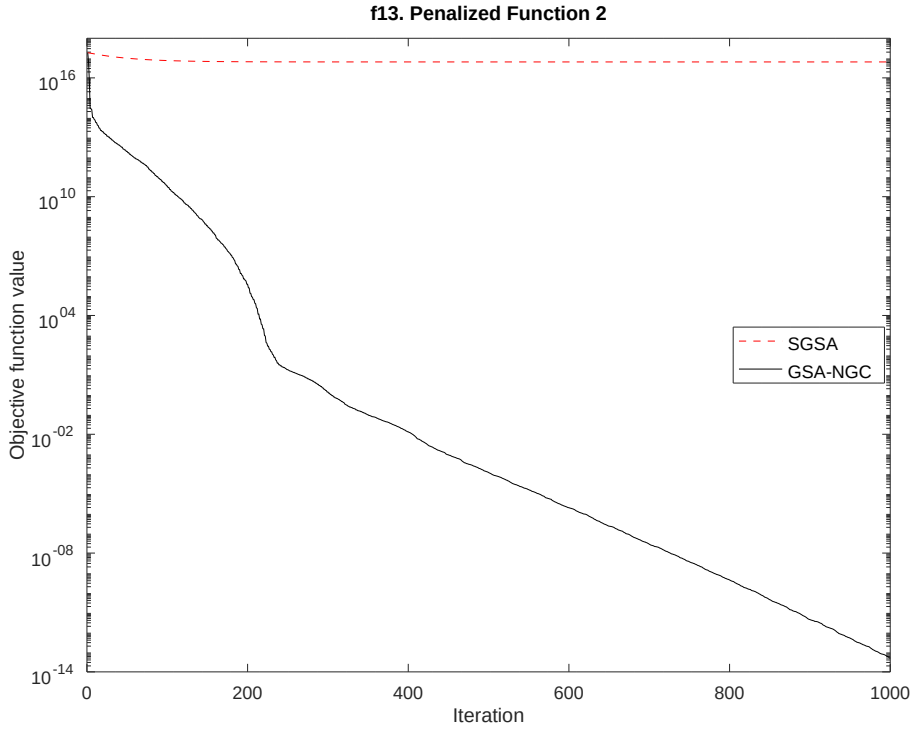

Figura 87 – A função de referência 13 avaliada em um espaço de busca grande

APÊNDICE H – As funções de referência avaliadas em um espaço de busca irregular

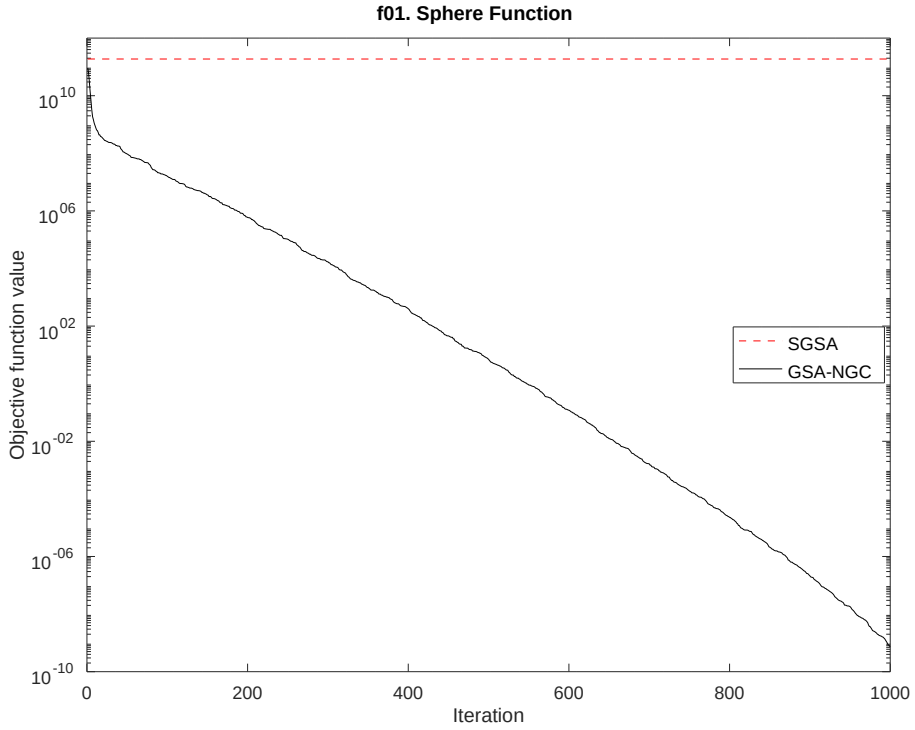

Figura 88 – A função de referência 1 avaliada em um espaço de busca irregular

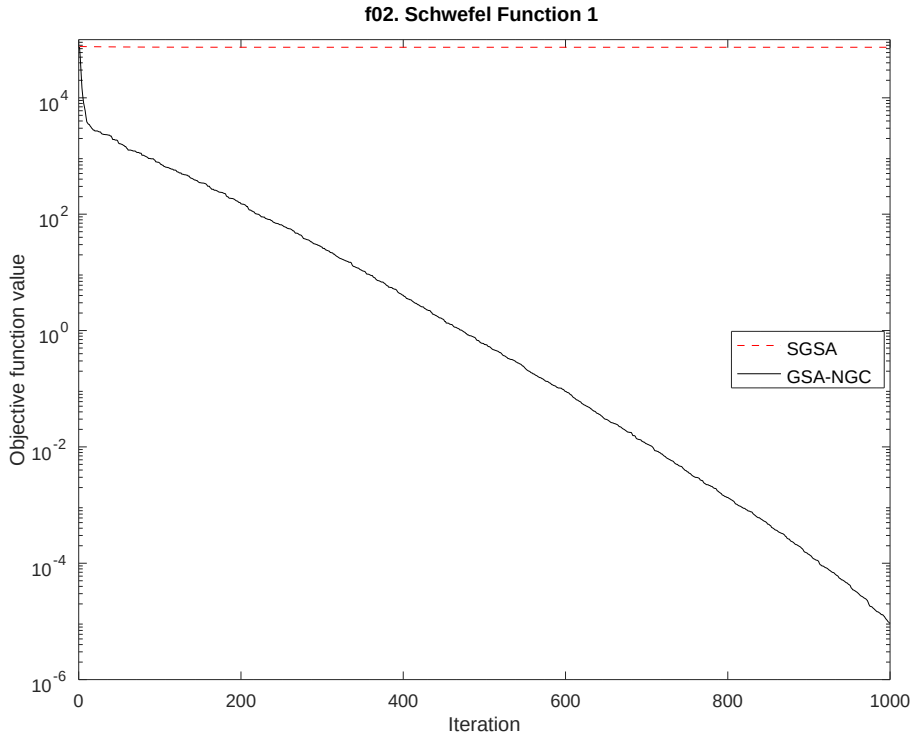

Figura 89 – A função de referência 2 avaliada em um espaço de busca irregular

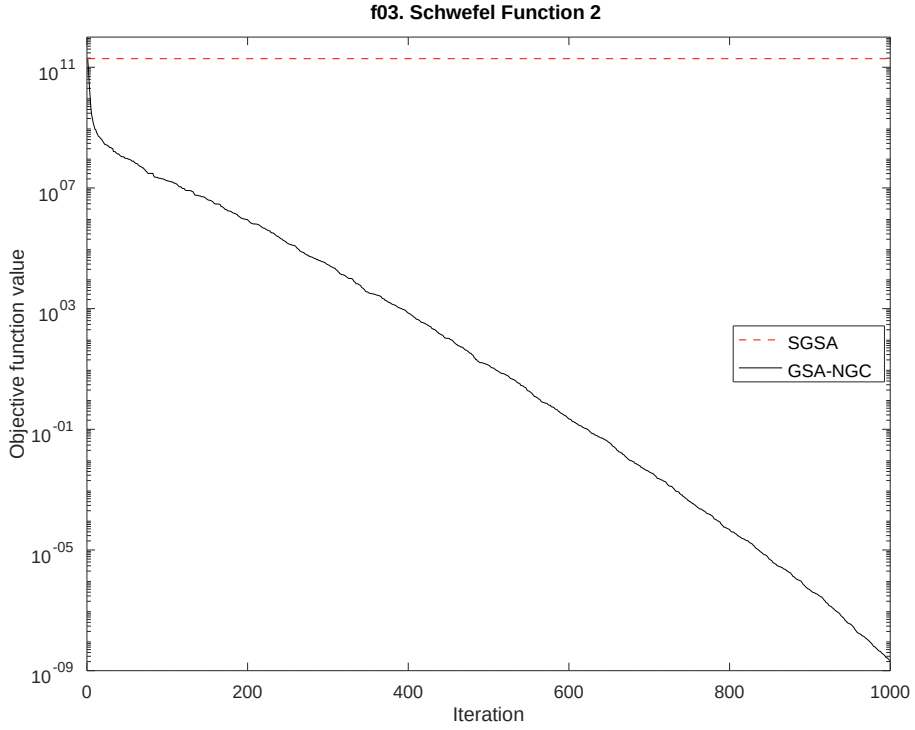

Figura 90 – A função de referência 3 avaliada em um espaço de busca irregular

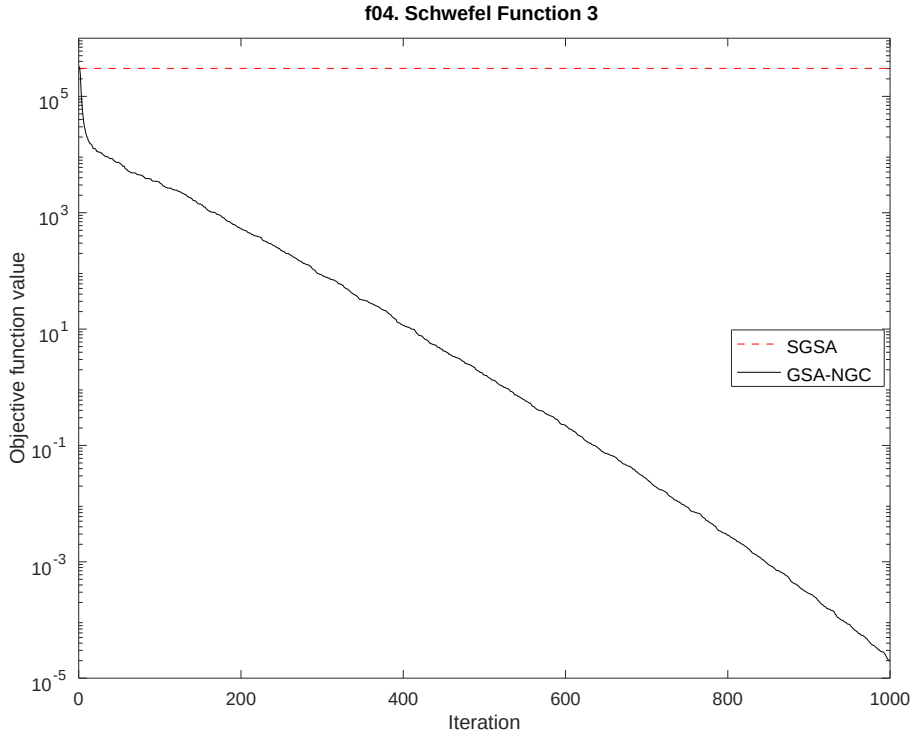

Figura 91 – A função de referência 4 avaliada em um espaço de busca irregular

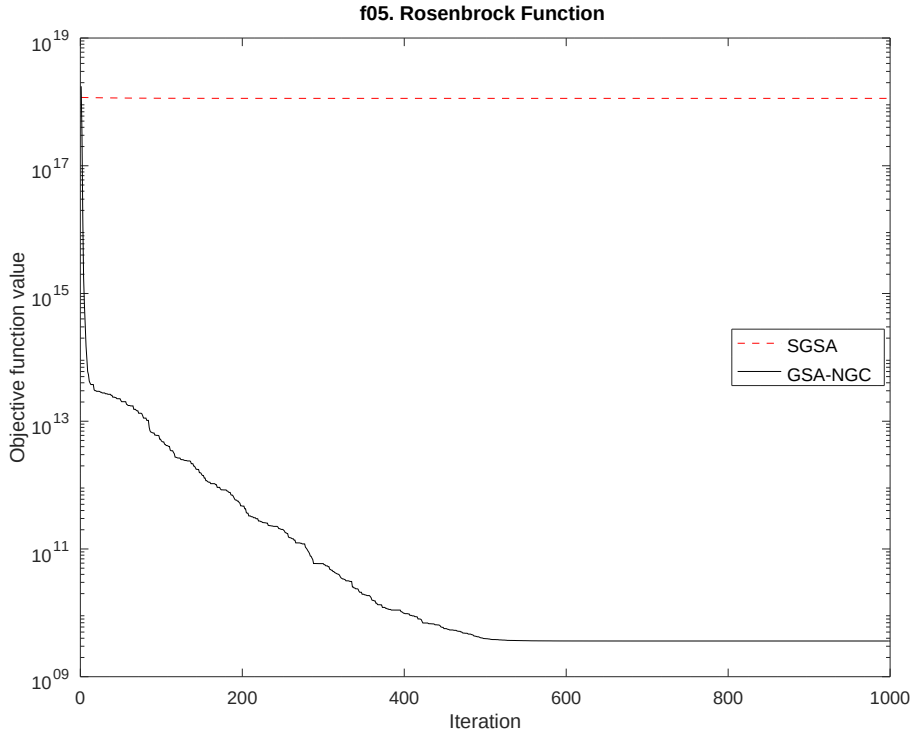

Figura 92 – A função de referência 5 avaliada em um espaço de busca irregular

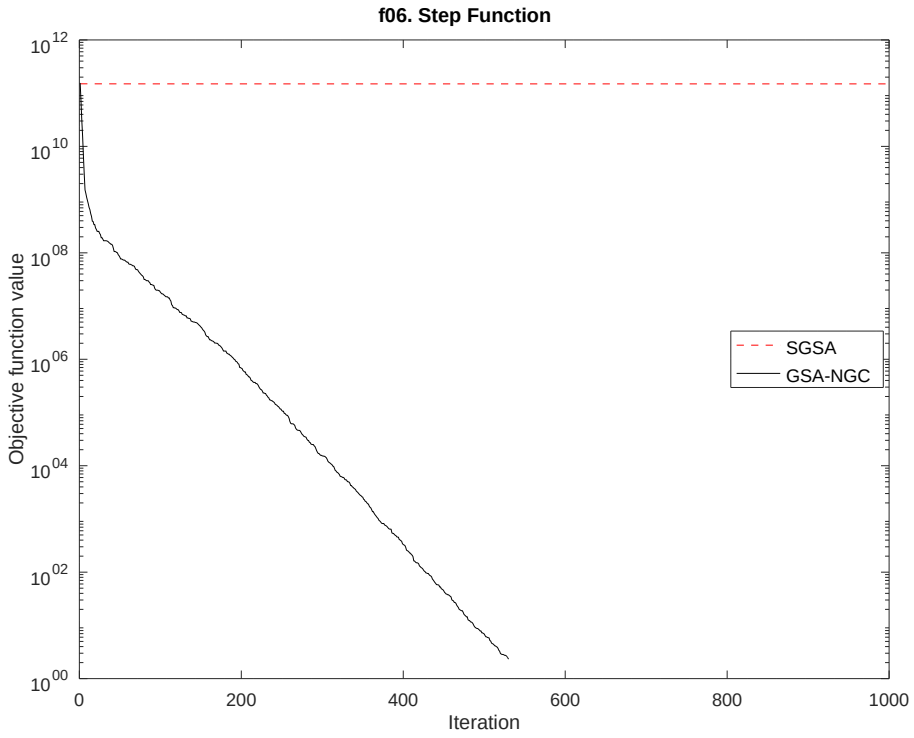

Figura 93 – A função de referência 6 avaliada em um espaço de busca irregular

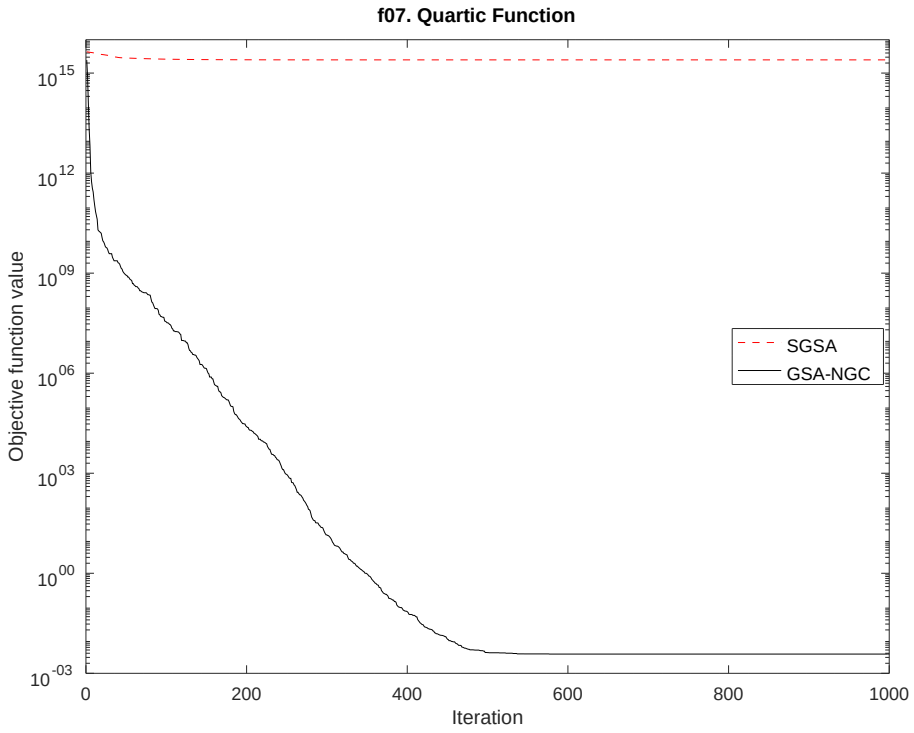

Figura 94 – A função de referência 7 avaliada em um espaço de busca irregular

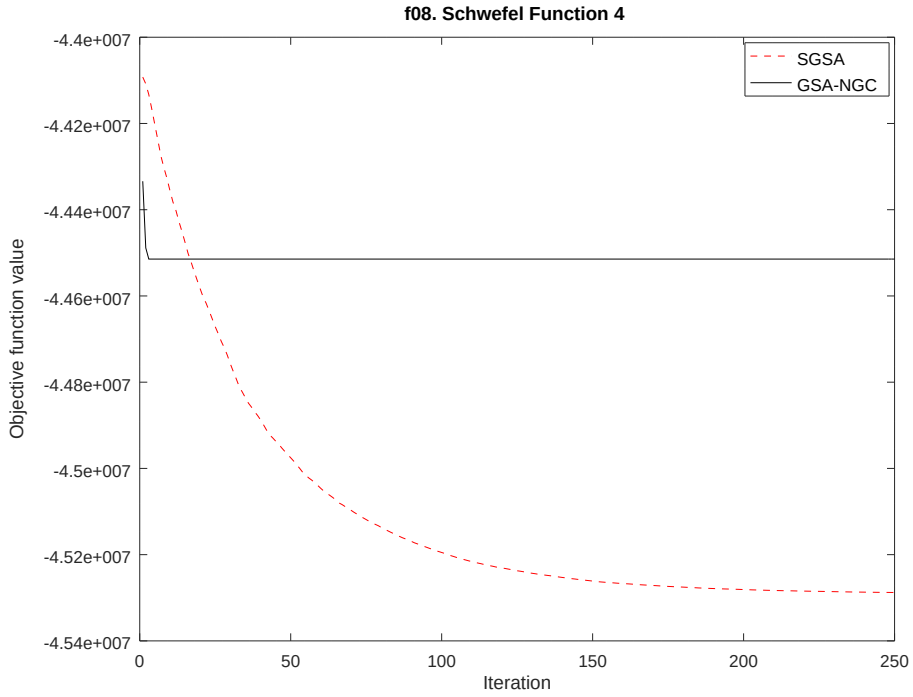

Figura 95 – A função de referência 8 avaliada em um espaço de busca irregular

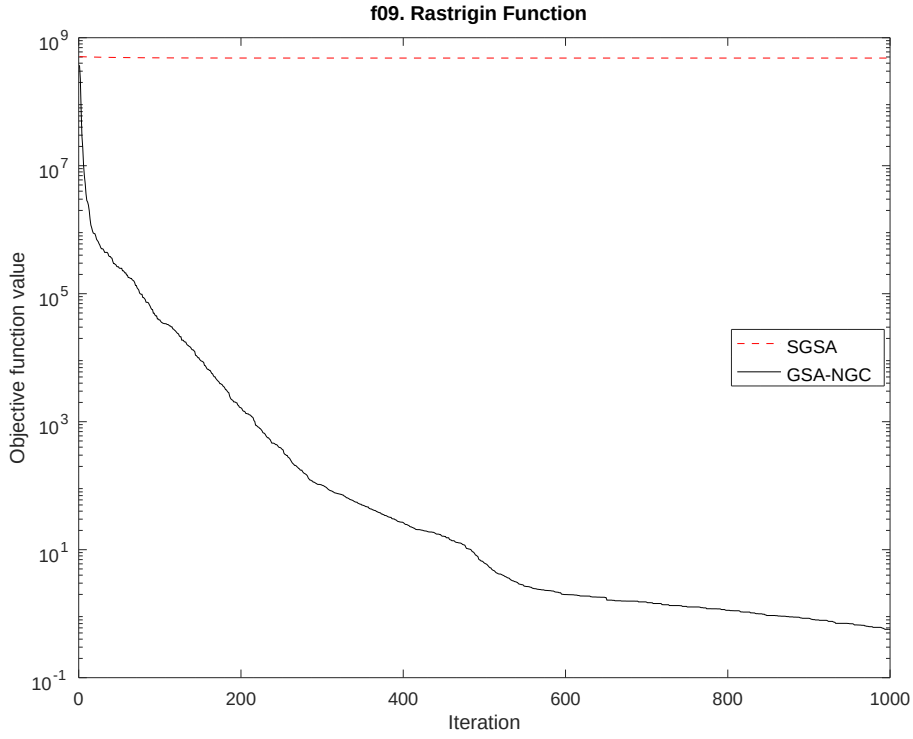

Figura 96 – A função de referência 9 avaliada em um espaço de busca irregular

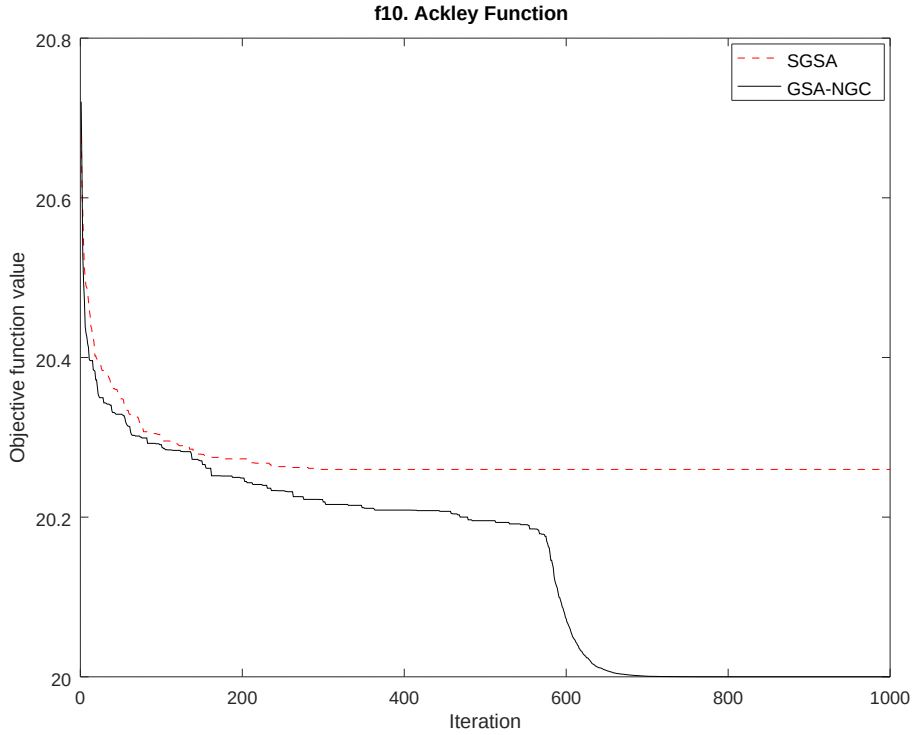

Figura 97 – A função de referência 10 avaliada em um espaço de busca irregular

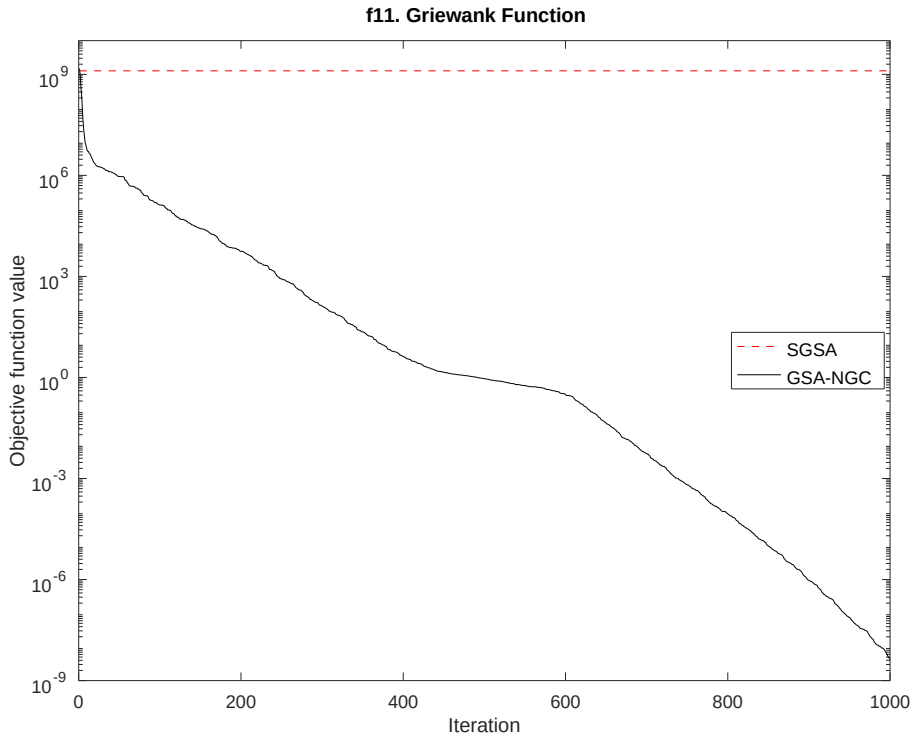

Figura 98 – A função de referência 11 avaliada em um espaço de busca irregular

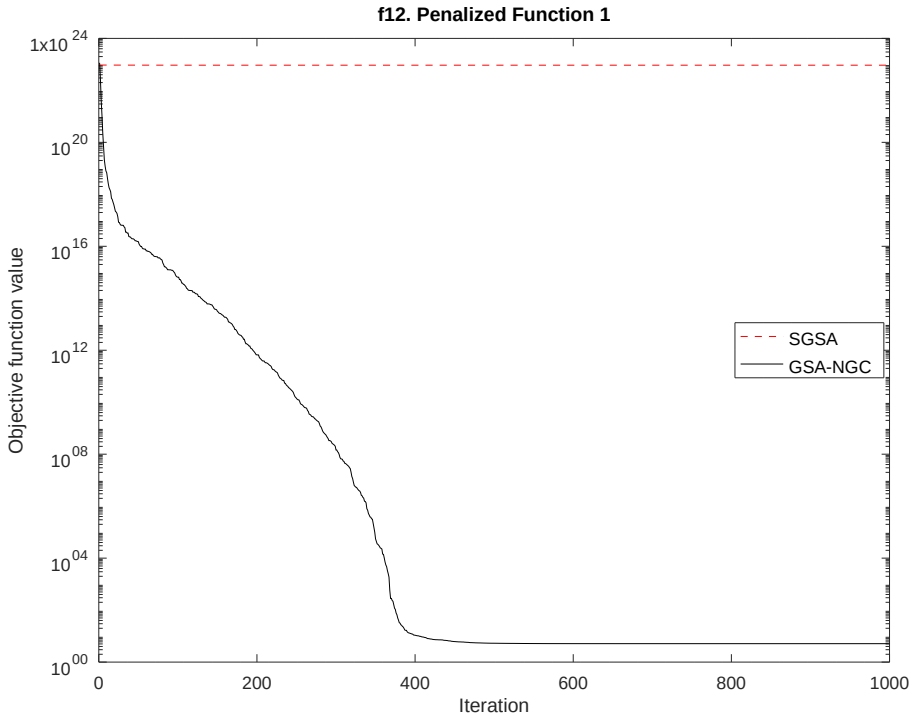

Figura 99 – A função de referência 12 avaliada em um espaço de busca irregular

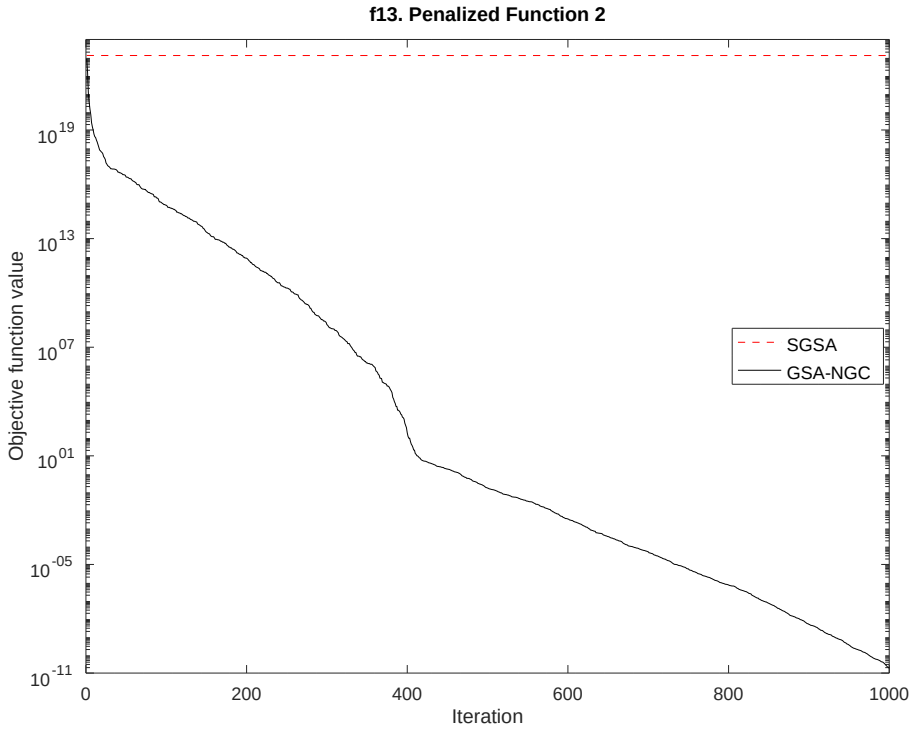

Figura 100 – A função de referência 13 avaliada em um espaço de busca irregular

Referências

BANSAL, J. C.; JOSHI, S. K.; NAGAR, A. K. Fitness varying gravitational constant in gsa. *Applied Intelligence*, 2018.

BEIGVAND, S. D.; ABDI, H.; SCALA, M. L. Hybrid gravitational search algorithm-particle swarm optimization with time varying acceleration coefficients for large scale chped problem. *Energy*, v. 126, p. 841–853, 2017.

BOUSSAïD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. *Information Sciences*, v. 237, p. 82–117, 2013.

BRANS, C.; DICKE, R. H. Mach's principle and a relativistic theory of gravitation. *Physical Review*, v. 124, n. 3, 1961.

CHEN, F.; ZHOU, J.; WANG, C.; LI, C.; LU, P. A modified gravitational search algorithm based on a non-dominated sorting genetic approach for hydro-thermal-wind economic emission dispatching. *Energy*, v. 121, p. 276–291, 2017.

D'INVERNO, R. *Introducing Einstein's Relativity*. [S.l.]: Oxford University Press, 1992.

DORAGHINEJAD, M.; NEZAMABADI-POUR, H. Black hole: A new operator for gravitational search algorithm. *International Journal of Computational Intelligence Systems*, v. 7, p. 809–826, 2014.

DORAGHINEJAD, M.; NEZAMABADI-POUR, H.; HASHEMPOURSADEGHIAN, A.; MAGHFOORI, M. A hybrid algorithm based on gravitational search algorithm for unimodal optimization. *International eConference on Computer and Knowledge Engineering*, p. 129–132, 2012.

DOWLATSHAHI, M. B.; NEZAMABADI-POUR, H.; MASHINCHI, M. A discrete gravitational search algorithm for solving combinatorial optimization problems. *Information Sciences*, v. 258, p. 94–107, 2014.

DUMAN, S.; GüVENç, U.; SöNMEZ, Y.; YöRüKEREN, N. Optimal power flow using gravitational search algorithm. *Energy Conversion and Management*, v. 59, p. 86–95, 2012.

DUMAN, S.; GüVENç, U.; YöRüKEREN, N. Gravitational search algorithm for economic dispatch with valve-point effects. *International Review of Electrical Engineering*, v. 5, n. 6, p. 2890–2895, 2010.

GOLDBERG, D. E.; DEB, K. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 1991.

HASHEMI, Y.; SHAYEGHI, H.; MORADZADEH, M. Design of dual-dimensional controller based on multi-objective gravitational search optimization algorithm for amelioration of impact of oscillation in power generated by large-scale wind farms. *Applied Soft Computing*, v. 53, p. 236–261, 2017.

HASSANZADEH, H. R.; ROUHANI, M. A multi-objective gravitational search algorithm. *International Conference on Computational Intelligence, Communication Systems and Networks*, p. 7–12, 2010.

HOLLAND, J. H.; GOLDBERG, D. E. Genetic algorithms and machine learning. *Machine Learning*, v. 3, p. 95–99, 1988.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. *Encyclopedia of Machine Learning*, p. 1942–1948, 1995.

KHAJOOEI, F.; RASHEDI, E. A new version of gravitational search algorithm with negative mass. *Conference on Swarm Intelligence and Evolutionary Computation*, p. 1–5, 2016.

LI, C.; ZHANG, N.; LAI, X.; ZHOU, J.; XU, Y. Design of a fractional-order pid controller for a pumped storage unit using a gravitational search algorithm based on the cauchy and gaussian mutation. *Information Sciences*, v. 396, p. 162–181, 2017.

LI, C.; ZHOU, J. Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. *Energy Conversion Management*, v. 52, p. 374–381, 2011.

LUENBERGER, D. G.; YE, Y. *Linear and Nonlinear Programming*. 4. ed. [S.l.]: Springer Publishing, 2016.

LUKE, S. *Essentials of Metaheuristics*. 2. ed. [S.l.: s.n.], 2009. Disponível online: <https://cs.gmu.edu/~sean/book/metaheuristics/>.

MACH, E. *The Science of Mechanics*. 3. ed. [S.l.]: The Open Court Publishing Company, 1919.

MANSOURI, R.; NASSERI, F.; KHORRAMI, M. Effective time variation of g in a model universe with variable space dimension. *Physics Letters A*, v. 259, p. 194–200, 1999.

MARZBANDA, M.; GHADIMIB, M.; SUMPER, A.; DOMíNGUEZ-GARCíA, J. L. Experimental validation of a real-time energy management system using multi-period gravitational search algorithm for microgrids in islanded mode. *Applied Energy*, v. 128, p. 164–174, 2014.

MIRJALILI, S.; HASHIM, S. Z. M.; SARDROUDI, H. M. Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*, v. 218, p. 11125–11137, 2012.

MITCHELL, M. *An Introduction to Genetic Algorithms*. 5. ed. [S.l.]: MIT Press, 1999.

MOEINI, R.; SOLTANI-NEZHAD, M.; DAEI, M. Constrained gravitational search algorithm for large scale reservoir operation optimization problem. *Engineering Applications of Artificial Intelligence*, v. 62, p. 222–233, 2017.

NEWTON, I. *Mathematical Principles of Natural Philosophy*. 1. ed. [S.l.]: Daniel Adee, 1846.

PACKIASUDHA, M.; SUJA, S.; JEROME, J. A new cumulative gravitational search algorithm for optimal placement of fact device to minimize system loss in the deregulated electrical power environment. *Electrical Power and Energy Systems*, v. 84, p. 34–46, 2017.

RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. Gsa: A gravitational search algorithm. *Information Sciences*, v. 179, p. 2232–2248, 2009.

RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. Bgsa: Binary gravitational search algorithm. *Natural Computing*, v. 9, p. 727–745, 2010.

RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, v. 24, p. 117–122, 2011.

SABRI, N. M.; PUTEH, M.; MAHMOOD, M. R. A review of gravitational search algorithm. *Internetional Journal of Advances in Soft Computing and its Application*, v. 5, n. 3, 2013.

SAEIDI-KHABISI, F. sadat; RASHEDI, E. Fuzzy gravitational search algorithm. *International eConference on Computer and Knowledge Engineering*, p. 156–160, 2012.

SARAFRAZI, S.; NEZAMABADI-POUR, H.; SARYAZDI, S. Disruption: A new operator in gravitational search algorithm. *Scientia Iranica*, v. 18, p. 539–548, 2011.

STRONG, J. *Concordância Exaustiva da Bíblia*. [S.l.]: Sociedade Bíblica do Brasil, 2002.

WEISE, T. *Global Optimization Algorithms: Theory and Applications*. 2. ed. [S.l.]: Free Software Foundation, 2009.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for search. *Santa Fe Institution Publications*, 1996.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1997.

YAZDANI, S.; NEZAMABADI-POUR, H.; KAMYAB, S. A gravitational search algorithm for multimodal optimization. *Swarm and Evolutionary Computation*, v. 14, p. 1–14, 2014.