Pontifícia Universidade Católica
do Rio de Janeiro

Manoela Rabello Kohler

PSO+: Algoritmo com Base em Enxame de Partículas para Problemas com Restrições Lineares e Não Lineares

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-Graduação em Engenharia Elétrica da PUC-Rio.

Orientador: Prof. Ricardo Tanscheit
Co-Orientadora: Profa. Marley Maria Bernardes Rebuzzi Vellasco

Rio de Janeiro
Setembro de 2017

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Manoela Rabello Kohler

PSO+: Algoritmo com Base em Enxame de Partículas para Problemas com Restrições Lineares e Não Lineares

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Ricardo Tanscheit
Orientador
Departamento de Engenharia Elétrica – PUC-Rio

Profa. Marley Maria Bernardes Rebuzzi Vellasco
Co-orientadora
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Marco Aurélio Cavalcanti Pacheco
Departamento de Engenharia Elétrica – PUC-Rio

Profa. Sandra Aparecida Sandri
Instituto Nacional de Pesquisas Espaciais - INPE

Prof. Jorge Luís Machado do Amaral
Universidade do Estado do Rio de Janeiro - UERJ

Prof. Douglas Mota Dias
Universidade do Estado do Rio de Janeiro – UERJ

Prof. Marco Antônio Cardoso
Cenpes - Petrobras

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 5 de Setembro de 2017

Manoela Rabello Kohler

Graduou-se em Engenharia de Computação pela Pontifícia Universidade Católica Do Rio de Janeiro (PUC-Rio) em 2011. Mestre em Engenharia Elétrica – área de concentração: Métodos de Apoio à Decisão – pela Pontifícia Universidade Católica Do Rio de Janeiro (PUC-Rio) em 2013. Trabalha como pesquisadora e desenvolvedora em projetos de sistemas de apoio à decisão.

Ficha Catalográfica

Kohler, Manoela Rabello

PSO+: algoritmo com base em enxame de partículas para problemas com restrições lineares e não lineares / Manoela Rabello Kohler; orientador: Ricardo Tanscheit; co-orientadora: Marley Maria Bernardes Rebuzzi Vellasco – 2017.

132 f.: il. (color.) ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2017.

Inclui referências bibliográficas.

1. Engenharia Elétrica – Teses. 2. Otimização. 3. Inteligência de Enxames. 4. Enxame de Partículas. 5. Restrições Lineares. 6. Restrições Não Lineares. I. Tanscheit, Ricardo. II. Vellasco, Marley Maria Bernardes Rebuzzi. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

Agradecimentos

Aos meus orientadores: Professor Ricardo e Professora Marley, pelo apoio para a realização deste trabalho.

Aos meus pais, pela educação, atenção e carinho.

Aos meus irmãos amigos e amigos irmãos, por todo apoio, paciência e compreensão.

A todos os amigos da PUC-Rio.

Aos professores que participaram da Comissão examinadora.

Ao CNPq, à PUC-Rio, e à Petrobras pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Resumo

Kohler, Manoela Rabello; Tanscheit, Ricardo; Vellasco, Marley Maria Bernardes Rebuzzi. **PSO+: Algoritmo com Base em Enxame de Partículas para Problemas com Restrições Lineares e Não Lineares.** Rio de Janeiro, 2017. 132p. Tese de Doutorado - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

O algoritmo de otimização por enxame de partículas (PSO, do inglês *Particle Swarm Optimization*) é uma meta-heurística baseada em populações de indivíduos na qual os candidatos à solução evoluem através da simulação de um modelo simplificado de adaptação social. Juntando robustez, eficiência e simplicidade, o PSO tem adquirido grande popularidade. São reportadas muitas aplicações bem-sucedidas do PSO nas quais este algoritmo demonstrou ter vantagens sobre outras meta-heurísticas bem estabelecidas baseadas em populações de indivíduos. Algoritmos modificados de PSO já foram propostos para resolver problemas de otimização com restrições de domínio, lineares e não lineares. A grande maioria desses algoritmos utilizam métodos de penalização, que possuem, em geral, inúmeras limitações, como por exemplo: (*i*) cuidado adicional ao se determinar a penalidade apropriada para cada problema, pois deve-se manter o equilíbrio entre a obtenção de soluções válidas e a busca pelo ótimo; (*ii*) supõem que todas as soluções devem ser avaliadas. Outros algoritmos que utilizam otimização multi-objetivo para tratar problemas restritos enfrentam o problema de não haver garantia de se encontrar soluções válidas. Os algoritmos PSO propostos até hoje que lidam com restrições, de forma a garantir soluções válidas utilizando operadores de viabilidade de soluções e de forma a não necessitar de avaliação de soluções inválidas, ou somente tratam restrições de domínio controlando a velocidade de deslocamento de partículas no enxame, ou o fazem de forma ineficiente, reinicializando aleatoriamente cada partícula inválida do enxame, o que pode tornar inviável a otimização de determinados problemas. Este trabalho apresenta um novo algoritmo de otimização por enxame de partículas, denominado PSO+, capaz de resolver problemas com restrições lineares e não lineares de forma a solucionar essas deficiências. A modelagem do algoritmo agrega seis diferentes capacidades para resolver problemas de otimização com restrições: (*i*) redirecionamento

aritmético de validade de partículas; (*ii*) dois enxames de partículas, onde cada enxame tem um papel específico na otimização do problema; (*iii*) um novo método de atualização de partículas para inserir diversidade no enxame e melhorar a cobertura do espaço de busca, permitindo que a borda do espaço de busca válido seja devidamente explorada – o que é especialmente conveniente quando o problema a ser otimizado envolve restrições ativas no ótimo ou próximas do ótimo; (*iv*) duas heurísticas de criação da população inicial do enxame com o objetivo de acelerar a inicialização das partículas, facilitar a geração da população inicial válida e garantir diversidade no ponto de partida do processo de otimização; (*v*) topologia de vizinhança, denominada 'vizinhança de agrupamento aleatório coordenado' para minimizar o problema de convergência prematura da otimização; (*vi*) módulo de transformação de restrições de igualdade em restrições de desigualdade. O algoritmo foi testado em vinte e quatro funções *benchmarks* – criadas e propostas em uma competição de algoritmos de otimização –, assim como em um problema real de otimização de alocação de poços em um reservatório de petróleo. Os resultados experimentais mostram que o novo algoritmo é competitivo, uma vez que aumenta a eficiência do PSO e a velocidade de convergência.

Palavras-chave

Otimização; Inteligência de Enxames; Enxame de Partículas; Restrições Lineares; Restrições Não Lineares; PSO.

Abstract

Kohler, Manoela Rabello; Tanscheit, Ricardo (Advisor); Vellasco, Marley Maria Bernardes Rebuzzi (Co-Advisor). **PSO+: A Linear and Nonlinear Constraints-Handling Particle Swarm Optimization.** Rio de Janeiro, 2017. 132p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

The Particle Swarm Optimization (PSO) algorithm is a metaheuristic based on populations of individuals in which solution candidates evolve through simulation of a simplified model of social adaptation. By aggregating robustness, efficiency and simplicity, PSO has gained great popularity. Many successful applications of PSO are reported in which this algorithm has demonstrated advantages over other well-established metaheuristics based on populations of individuals. Modified PSO algorithms have been proposed to solve optimization problems with domain, linear and nonlinear constraints; The great majority of these algorithms make use of penalty methods, which have, in general, numerous limitations, such as: (i) additional care in defining the appropriate penalty for each problem, since a balance must be maintained between obtaining valid solutions and the searching for an optimal solution; (ii) they assume all solutions must be evaluated. Other algorithms that use multi-objective optimization to deal with constrained problems face the problem of not being able to guarantee finding feasible solutions. The proposed PSO algorithms up to this date that deal with constraints, in order to guarantee valid solutions using feasibility operators and not requiring the evaluation of infeasible solutions, only treat domain constraints by controlling the velocity of particle displacement in the swarm, or do so inefficiently by randomly resetting each infeasible particle, which may make it infeasible to optimize certain problems. This work presents a new particle swarm optimization algorithm, called PSO+, capable of solving problems with linear and nonlinear constraints in order to solve these deficiencies. The modeling of the algorithm has added six different capabilities to solve constrained optimization problems: (i) arithmetic redirection to ensure particle feasibility; (ii) two particle swarms, where each swarm has a specific role in the optimization the problem; (iii) a new particle updating method to insert diversity into the swarm and improve the coverage of the

search space, allowing its edges to be properly exploited – which is especially convenient when the problem to be optimized involves active constraints at the optimum solution; (iv) two heuristics to initialize the swarm in order to accelerate and facilitate the initialization of the feasible initial population and guarantee diversity at the starting point of the optimization process; (v) neighborhood topology, called coordinated random clusters neighborhood to minimize optimization premature convergence problem; (vi) transformation of equality constraints into inequality constraints. The algorithm was tested for twenty-four benchmark functions – created and proposed for an optimization competition – as well as in a real optimization problem of well allocation in an oil reservoir. The experimental results show that the new algorithm is competitive, since it increases the efficiency of the PSO and the speed of convergence.

Keywords

Optimization; Swarm Intelligence; Particle Swarm; Linear Constraints; Nonlinear Constraints; PSO.

Sumário

Lista de Figuras

Lista de Tabelas

1.
Introdução

1.1.
Motivações

A otimização de funções com restrições é uma área de particular importância em aplicações de engenharia, matemática, economia, entre outras. Restrições podem tanto representar limites físicos de sistemas quanto intervalos desejáveis de operação (Vega-Alvarado *et al.*, 2017).

O objetivo da otimização com restrições é a busca de soluções viáveis com os melhores valores objetivos (He e Wang, 2007). De um modo geral, um problema de otimização restrita pode ser descrito como:

Encontrar *x* para minimizar (ou maximizar) $f(\vec{x})$

Sujeito à $\boldsymbol{g_i(\vec{x}) \leq 0, i = 1, 2, \dots, n,}$

$$\boldsymbol{h_j(\vec{x}) = 0, j = 1, 2, \dots, p,} \qquad (1)$$

onde $\vec{x} = [x_1, x_2, \dots, x_d]$ denota o vetor solução, *n* é o número de restrições de desigualdade e *p* é o número de restrições de igualdade. Na prática, uma restrição de igualdade $h_j(x) = 0$ pode ser substituída por um par de restrições de desigualdade $h_j(x) \leq \delta$ e $h_j(x) \geq \delta$ (δ é um pequeno valor de tolerância). Portanto, todas as restrições podem ser transformadas em $N = n + 2p$ restrições de desigualdade. O espaço de busca *S* é definido pelas restrições de domínio de cada variável da solução $\vec{x}$. Se *S* é um conjunto discreto, o problema é combinatório; se é contínuo, o problema é numérico (Dominguez-Isidro, Mezura-Montes e Leguizamon, 2013). Uma solução $\vec{x}$ é chamada de solução viável desde que respeite todas as restrições.

Existe uma classe de problemas, particularmente em aplicações do mundo real, onde é impossível encontrar uma solução ótima usando uma quantidade viável de recursos quando técnicas clássicas, tais como métodos numéricos ou análise gráfica, são utilizadas. Estes casos correspondem à categoria *hard* de otimização e têm uma natureza semelhante aos problemas de decisão de tempo polinomial não

determinístico (NP), uma vez que não podem ser resolvidos de maneira ótima utilizando métodos determinísticos em um tempo polinomial. Meta-heurísticas são algoritmos de otimização desenvolvidos para resolver este tipo de problemas de modo aproximado por técnicas de tentativa e erro, sem exigir uma adaptação profunda para cada caso (Boussaïd, Lepagnot e Siarry, 2013).

Com base na simulação de modelos sociais simplificados, como o comportamento de um bando de pássaros em revoada e de cardumes de peixes, a Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO) foi introduzida pela primeira vez (Kennedy e Eberhart, 1995) como uma nova técnica de computação evolucionária com o mecanismo de melhoria individual, da cooperação da população e da concorrência. A otimização com PSO tem sido utilizada em vários problemas de otimização em anos recentes (Abdessamia *et al.*, 2017; Hoang *et al.*, 2017; Peimankar *et al.*, 2017; Qolomany *et al.*, 2017; Raitoharju *et al.*, 2017; Rey e Zmeureanu, 2017). No PSO, vários indivíduos candidatos, chamados partículas, convivem e colaboram entre si. A posição de cada partícula denota um vetor de decisão para o problema original. A trajetória de cada partícula no espaço de busca é ajustada dinamicamente, atualizando-se a sua velocidade de acordo com a sua própria experiência de "voo", bem como com a experiência de partículas vizinhas (construída através de rastreamento e memorização da melhor posição encontrada até o momento). Portanto, o PSO combina a técnica de busca local (pela própria experiência da partícula) e o método de pesquisa global (pela experiência vizinha) para assim equilibrar a prospecção e exploração e, finalmente, alcançar o ótimo global (Chang e Shih, 2010).

Nos últimos anos, as técnicas de inteligência de enxames têm experimentado um crescimento, tanto no tocante à multiplicidade de algoritmos em desenvolvimento quanto na variedade das aplicações resultantes. Este crescimento é devido, em parte, a fatores como o grande número de problemas que envolvem características como não-linearidade, alta dimensionalidade, dificuldade de modelagem, impossibilidade no cálculo de derivadas, entre outros. Os algoritmos de enxame, de modo geral, têm mostrado resultados promissores em termos de robustez de descoberta de soluções, da velocidade da solução e precisão numérica. Grande parte desse êxito é também devido à simplicidade, versatilidade, generalidade e paralelismo desses algoritmos, sobretudo quando comparados

quanto à dificuldade de modelagem matemática de certos problemas (Poli, Kennedy e Blackwell, 2007).

Algoritmos evolucionários capazes de lidar com restrições podem cair em diferentes áreas (Michalewicz, 1995), dentre elas: (*i*) métodos baseados em funções de penalidade; (*ii*) métodos baseados na viabilidade de soluções.

A função de penalização tem sido o método de controle de restrição mais popular devido à sua simplicidade e facilidade de implementação. As violações das restrições das soluções são incorporadas na função objetivo para que os problemas originais com restrição sejam transformados em problemas sem restrições. Um método simples para se penalizar soluções não-válidas é aplicar uma penalidade constante. Assim, a função objetivo penalizada assume o valor da função objetivo original adicionada da penalidade. A função de penalização para o problema de minimização com restrições pode ser escrita da seguinte forma:

$$\boldsymbol{F_p = F + \sum C_i \delta_i} \qquad (2)$$

onde F_p é a função objetivo penalizada, F é a função objetivo original, C_i é o coeficiente de penalização imposto para a violação de i-ésima restrição e δ_i indica se a restrição foi satisfeita (= 0) ou violada (= 1).

Diversos autores já propuseram novos algoritmos com penalização, tendo como base o PSO, que podem ser aplicados a problemas com restrições. O algoritmo PSO baseado em multiagente evolucionário apresentado em (Kumar, Anand e Sydulu, 2011) resolve o problema de fluxo de potência ótimo com restrições de segurança que geram termos de penalização para a função objetivo. Em (Vega-Alvarado *et al.*, 2017), os autores propõem um algoritmo híbrido de Colônia de Abelhas e *Random Walk* para tratar um problema real de engenharia. Para tratar as restrições inerentes ao problema, é proposta uma função de penalização para soluções inválidas e posteriormente é utilizado um operador de competição para escolher entre todas as soluções quais são as melhores: (*i*) entre duas soluções válidas, a melhor será a que obtiver melhor resposta da função de avaliação; (*ii*) entre uma solução válida e uma inválida, a solução válida é considerada melhor; (*iii*) entre duas soluções inválidas, é escolhida a que tiver violado menos restrições. Em (Saha *et al.*, 2016) é proposta uma nova abordagem baseada em regras *fuzzy* para solucionar problemas com restrições não lineares. Em (Li, Tian e Kong, 2005), um novo PSO para resolver problemas de otimização com

restrições é proposto. Um fator de penalidade é introduzido para garantir que a avaliação de soluções inviáveis seja sempre pior do que qualquer solução viável. Em (Xu, Lei e Sun, 2010), os autores propõem três algoritmos PSO diferentes para melhorar a cobertura de redes de câmeras. Um dos algoritmos adiciona uma penalidade adaptável à cobertura quando uma câmara viola a restrição de distância. (Jie Li, Sava e Xiaolan Xie, 2009) propuseram um método de busca aleatória baseado em uma busca local e introduziu uma função de penalização para lidar com as restrições. Esse método de penalização foi estendido em (Park e Kim, 2015), em que se define um parâmetro diferente de penalização chamado 'Penalização em Sequência', com bom desempenho em solução na borda do espaço válido.

Funções de penalização têm, em geral, inúmeras limitações. Particularmente, elas não são uma boa opção para problemas em que a solução ótima se encontra na fronteira entre a região válida e a região inválida ou quando as regiões válidas são disjuntas (Liang *et al.*, 2006). Além disso, funções de penalização necessitam de um cuidado adicional ao se determinar a penalidade apropriada para determinado problema, pois enfrentam a dificuldade de manter um equilíbrio entre a obtenção de soluções válidas e a busca pelo ótimo. Uma penalidade demasiadamente elevada pode forçar o algoritmo a encontrar uma solução válida ainda que não ótima. Por outro lado, se a penalidade for pequena, a ênfase na validade é reduzida e o sistema pode nunca convergir. Além disso, os métodos supõem que todas as soluções devem ser avaliadas e não são apropriados para um espaço de busca relativamente grande ou quando o cálculo da função objetivo é muito custosa do ponto de vista computacional.

A preservação de viabilidade de soluções é distinta da abordagem da função de penalização. Soluções são reparadas ou descartadas se elas não satisfazem as restrições do problema. A reparação de soluções inválidas garante que sempre haverá soluções válidas a cada iteração e que soluções inválidas ou não serão avaliadas ou nunca terão uma avaliação melhor que qualquer solução válida.

Além disso, no caso do PSO, esta abordagem, idealmente, deve implementar operadores de validade, que definem métodos a serem aplicados no caso de a partícula "voar" para fora dos limites das restrições, a fim de transformar uma solução inválida em uma solução válida. Em (Robinson e Rahmat-Samii, 2004), os autores propõem um operador para limitar o vetor de velocidade do PSO. Eles propõem três métodos para lidar com condições de contorno: (i) *Absorbing Walls* -

quando uma partícula atinge o limite do espaço de solução em uma das dimensões, a velocidade naquela dimensão é zerada e a partícula é, finalmente, puxada de volta para o espaço de solução permitido. No sentido de fronteira, as "paredes" absorvem a energia das partículas que tentam escapar do espaço de solução; (ii) *Reflecting Walls* - quando uma partícula atinge a fronteira em uma das dimensões, o sinal da velocidade naquela dimensão é alterado e a partícula é refletida de volta para o espaço de solução válido; (iii) *Invisible Walls* - as partículas podem voar sem qualquer restrição física. No entanto, as partículas que circulam fora do espaço de solução válido não são avaliadas. Estes métodos apresentam bom desempenho com restrições de fronteira ou domínio – quando as restrições são impostas a cada variável individualmente –, mas não conseguem lidar com restrições lineares e não-lineares de igualdade e desigualdade em problemas de otimização.

Hu & Eberhart (2002) propõem um outro método em que a estratégia de preservação de viabilidade é empregada para lidar com restrições com PSO. Duas modificações para o PSO original são feitas: (*i*) durante a inicialização, todas as partículas são repetidamente iniciadas até que satisfaçam todas as restrições; (*ii*) ao calcular as melhores avaliações das partículas e da história do enxame, somente posições viáveis são consideradas. É mostrado que essa implementação do PSO resolve muitos problemas de otimização não-linear com restrições não lineares de desigualdade, mas há a necessidade de as partículas serem sempre aleatoriamente reiniciadas se não satisfizerem todas as restrições. Dependendo do problema, isso pode ser demorado e até mesmo impraticável. Em (Liu, 2008), o autor propõe uma nova técnica para tratar problemas não-lineares com restrições não lineares como um problema de otimização biobjetivo: um objetivo é o objetivo original do problema, e o outro é o grau de violação das restrições, que deve sempre ser minimizado. Essa abordagem também não garante que se obtenham sempre soluções válidas.

Este trabalho propõe um novo algoritmo PSO, chamado PSO+, que utiliza um operador aritmético de viabilidade e dois enxames, permitindo que o algoritmo proposto não necessite avaliar soluções inválidas e garanta que haja sempre um enxame cujas partículas respeitem completamente todas as restrições. Um novo método de atualização de partículas é também proposto para inserir diversidade no enxame e melhorar a cobertura do espaço de busca, permitindo que a borda do espaço de busca seja também explorada – o que é especialmente conveniente

quando o problema a ser otimizado envolve restrições ativas no ótimo global. Duas heurísticas de criação da população inicial do enxame são também propostas para acelerar a inicialização das partículas, de forma a facilitar a geração da população inicial válida e garantir diversidade no ponto de partida do processo de otimização. Além disso, é proposta uma topologia de vizinhança, denominada 'vizinhança de agrupamento aleatório coordenado' para minimizar o problema de convergência prematura da otimização. O algoritmo proposto é adequado para problemas de otimização lineares e não lineares com restrições lineares e não lineares, e também para problemas com restrições onde o custo de avaliação da função objetivo é muito alto, devendo-se então garantir que o enxame evolua sempre com soluções válidas, sem a necessidade de avaliação de possíveis soluções inválidas.

1.2. Objetivos

Os principais objetivos do presente trabalho são:

a) Propor e desenvolver um novo modelo baseado em PSO que trate problemas reais com restrições lineares e não lineares.
b) Propor e desenvolver um algoritmo para a obtenção de um conjunto de soluções iniciais que respeitem todas as restrições lineares e não lineares e que contenham maior diversidade.
c) O novo modelo deve garantir:
    - Soluções válidas
    - Exploração na borda do espaço de busca válido
    - Bom desempenho quando há restrições ativas no ótimo
    - Baixo tempo computacional
    - Diversidade das soluções no decorrer da otimização
    - Evitar a convergência prematura

1.3. Contribuições

As principais contribuições desse trabalho são:

- Desenvolvimento de um algoritmo evolucionário, tendo como base o PSO, que trate problemas reais com restrições lineares e não lineares:

  - Operador de cruzamento como método de garantir a validade das soluções;
  - Utilização de dois enxames, onde um deles respeita todas as restrições do problema para garantir a correta aplicação do operador de viabilidade de soluções, enquanto que o outro respeita somente restrições de domínio, garantindo a diversidade da otimização;
  - Criação do conceito de 'pontos de apoio' como forma de inserir diversidade na população, explorando melhor o espaço de busca e facilitando a exploração perto da borda do espaço de busca válido;
  - Topologia de vizinhança que minimize o problema da convergência prematura;
- Dois algoritmos de inicialização de soluções válidas e diversas, utilizando conceitos de otimização, agrupamento e Metropolis Hastings.

Contribuições secundárias:

- Aferição do desempenho de algumas características do PSO+ individualmente em *benchmarks* com restrições de domínio.
- Aferição do desempenho do modelo criado em *benchmarks* com estrições de domínio, lineares e não lineares conhecidos, comparando-o a outros algoritmos evolucionários.
- Avaliação do desempenho do modelo em um problema real de otimização com restrições físicas.

1.4. Organização da Tese

Este trabalho está estruturado da seguinte forma:

- O capítulo 2 apresenta os fundamentos da otimização por enxame de partículas, os mecanismos de funcionamento e as características dos principais algoritmos PSO para problemas com e sem restrições;
- O capítulo 3 apresenta os detalhes do algoritmo PSO+ proposto;

- O capítulo 4 apresenta os resultados obtidos com os vinte e quatro *benchmarks* e sua comparação com outros modelos, além da aplicação do modelo proposto em um caso real;
- O capítulo 5 apresenta as conclusões e trabalhos futuros.

2.
Otimização

O objetivo de uma otimização é chegar à solução mais eficiente para um dado problema. Na terminologia matemática, refere-se a métodos sistemáticos usados para encontrar os valores das variáveis, dentro dos limites do problema, que minimizam ou maximizam o valor da função objetivo. A melhor solução encontrada pode variar de um simples vetor que maximiza ou minimiza uma função até estruturas complexas, como o melhor arranjo de um time de robôs que leva ao maior número de gols em uma partida de futebol.

Atualmente, a área de otimização é composta por diversos campos, porém há algumas intersecções entre seus subcampos. Algumas subáreas são:

- Programação Linear: a função objetivo é linear e há um conjunto de restrições relacionadas às variáveis da função, que são especificadas como igualdades ou desigualdades lineares;
- Programação Inteira: apresenta problemas geralmente mais complexos que os de programação linear, com variáveis que se restringem a valores inteiros;
- Otimização Combinatória: o conjunto de soluções válidas é ou pode ser reduzido a um conjunto discreto;
- Otimização Estocástica: estuda os casos em que as restrições ou os parâmetros dependem de variáveis aleatórias.

2.1.
Conceitos Básicos

2.1.1.
Função Objetivo

A função objetivo é a representação matemática do critério de eficiência adotado no problema de otimização. É influenciada pelas variáveis de projeto,

conhecidas também como variáveis de controle do problema. Esse critério pode ser representado por uma única função objetivo, no caso de problemas mono-objetivo, ou por várias funções objetivo, no caso de problemas multiobjetivo. Da mesma forma, quando o critério de eficiência é influenciado por uma única variável, a função objetivo é dita unidimensional; e no caso em que várias variáveis tenham influência sobre o critério, a função objetivo é denominada multidimensional. Além disso, se a função tem um único mínimo (ou máximo), ela é chamada de função unimodal; quando a função tem vários mínimos (ou máximos), ela é denominada multimodal (Nariño, 2014).

2.1.2. Variáveis do Projeto

As variáveis de projeto são os parâmetros variáveis no processo de otimização que definem as características do modelo analisado. Em outras palavras, são as variáveis que se deseja determinar dentro de um intervalo predefinido e para as quais o problema é resolvido. De forma geral, as variáveis podem adotar qualquer valor; entretanto, algumas combinações de valores podem produzir soluções inviáveis no processo de otimização. Assim, é necessário definir as variáveis do projeto dentro de um intervalo cujos limites inferior e superior delimitem a sua variação.

Igualmente importante é a definição do tipo de variável – discreta ou contínua –, podendo existir problemas com ambos os tipos. Uma variável contínua é aquela que pode assumir qualquer valor dentro do intervalo de variação, enquanto que uma variável discreta é aquela que pode assumir valores específicos.

2.1.3. Restrições

As restrições são equações lineares ou não lineares, de igualdade ou de desigualdade, que determinam os limites de viabilidade do projeto, expressando uma condição desejável do comportamento do sistema. Geralmente, as restrições estão relacionadas com a geometria, os esforços admissíveis, os recursos disponíveis, os custos envolvidos, entre outros.

2.1.4.
Espaço de Busca

O espaço de busca é a região do domínio que satisfaz as restrições do problema. É também conhecido como domínio viável do problema. É delimitado pelas restrições impostas ao sistema e pelo intervalo de variação das variáveis do projeto. Podem existir "melhores soluções" fora do espaço de busca do problema, porém, tecnicamente, elas são inviáveis pelo fato de violarem as restrições.

2.1.5.
Ponto Ótimo

O ponto ótimo é o vetor das variáveis de projeto que minimiza (ou maximiza) a função objetivo e satisfaz as restrições do problema. O valor da função objetivo correspondente a essas variáveis é denominado valor ótimo. Consequentemente, o par formado pelo ponto ótimo e o valor ótimo é chamado de solução ótima, podendo ser local, se for um ponto de mínimo (ou máximo) em relação a uma vizinhança desse ponto, ou global, se for um valor mínimo (ou máximo) em relação a todo o espaço de busca do problema.

2.2.
Meta-heurísticas

As meta-heurísticas estão inclusas nos métodos de otimização estocástica e são algoritmos projetados para resolver de forma genérica uma variedade de problemas de otimização. Embora na prática seja impossível encontrar o valor ideal para todos os problemas usando esses métodos, a maioria dos casos produz soluções que são boas o suficiente para serem implementadas fisicamente. As características gerais de uma meta-heurística são: (*i*) tem componentes estocásticos envolvendo variáveis aleatórias; (*ii*) é inspirada em processos naturais ou artificiais; e (*iii*) requer uma série de parâmetros que precisam ser ajustados para o problema específico a ser resolvido (Boussaïd, Lepagnot e Siarry, 2013).

As meta-heurísticas baseadas em população lidam com um conjunto de indivíduos, cada um representando uma possível solução; dependendo da relação entre os indivíduos, esses métodos são classificados como computação evolucionária (EC, do inglês *Evolutionary Computation*) e/ou algoritmos de

inteligência de enxame (SIAs, do inglês *Swarm Intelligence Algorithms*) (Vega-Alvarado *et al.*, 2017). As meta-heurísticas mais conhecidas são: Algoritmo Genético (Hou *et al.*, 2017), Evolução Diferencial (Corne, Glover e Dorigo, 1999), Colônia de Formigas (Wang *et al.*, 2016), Colônia de Abelhas (Karaboga, 2005), e Enxame de Partículas (Kaur e Cheema, 2016).

2.3. Otimização por Enxame de Partículas

PSO é uma técnica baseada em inteligência de enxame e foi introduzida por Kennedy & Eberhart (1995). O comportamento do PSO pode ser imaginado comparando-o a enxames de aves em busca de fontes de alimento ideal, onde a direção em que um pássaro se move é influenciada por seu movimento atual, pela melhor fonte de alimento que já experimentou e pela melhor fonte de alimento que qualquer ave do enxame já experimentou. Em outras palavras, as aves são acionadas pela inércia, por seu conhecimento pessoal e pelo conhecimento do enxame. Em termos de PSO, o movimento de uma partícula é influenciado pela sua inércia, por sua melhor posição individual (*pbest*) e pela melhor posição global (*gbest*) ou pela melhor posição da sua vizinhança (*nbest*).

O indivíduo em um enxame é uma partícula sem volume em um espaço de busca multidimensional. A posição no espaço representa uma solução potencial para o problema de otimização. A velocidade de voo de uma partícula determina a direção e o passo da busca. A partícula voa no espaço de busca a uma velocidade definida, ajustada dinamicamente de acordo com a sua própria experiência de voo e com a experiência de outras partículas do enxame. A direção e a velocidade são constantemente ajustadas, traçando a melhor posição encontrada até o momento pelas próprias partículas e por todo o enxame. O enxame rastreia as duas melhores posições até o momento (*pbest* e *gbest ou nbest*), move-se gradualmente para a melhor região e, finalmente, tende a chegar à melhor posição de todo o espaço de busca (Lu *et al.*, 2008).

Em um PSO, cada partícula que representa uma solução potencial é mantida dentro de um enxame. Em termos simples, as partículas voam através de um espaço de busca multidimensional onde a posição de cada partícula é ajustada de acordo

com as suas experiências e a de seus vizinhos. Algumas implementações e evoluções do PSO são apresentadas a seguir.

2.3.1. Implementações de PSO

2.3.1.1. PSO Clássico

O PSO original (Kennedy e Eberhart, 1995) é descrito abaixo, onde a variável $x_i(t)$ denota a posição da partícula *i* no espaço de busca no passo *t*. A posição da partícula é alterada pela adição de um vetor conhecido como vetor de velocidade $v_i(t+1)$:

$$x_i(t+1) = x_i(t) + v_i(t+1) \qquad (3)$$

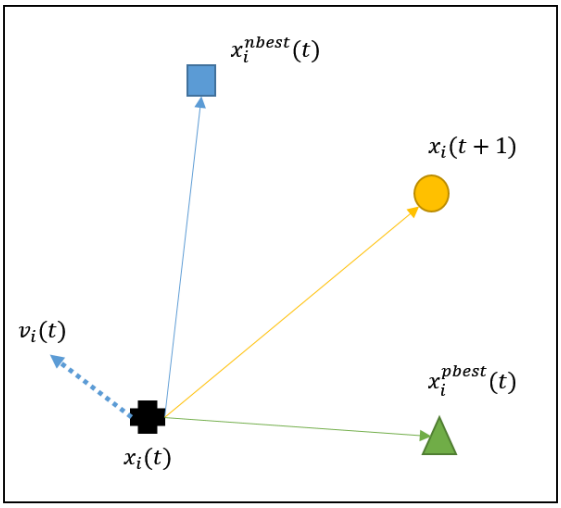
O processo de otimização é impulsionado pelo vetor de velocidade, que reflete o conhecimento da partícula (componente cognitivo) e informações socialmente trocadas com a vizinhança da partícula (componente social).

No PSO clássico, a velocidade de uma partícula é determinada pela seguinte equação:

$$v_i(t+1) = v_i(t) + 2 * rand_1 * \left(pbest_i(t) - x_i(t)\right) + 2 * rand_2 * (gbest(t) - x_i(t)) \qquad (4)$$

onde v_i é a velocidade da partícula *i* no instante *t*, $rand_1$ *e* $rand_2$ são valores aleatórios no intervalo [0,1] amostrados de uma distribuição uniforme contínua, $pbest$ denota a melhor posição da partícula *i* e $gbest$ denota a melhor posição global do enxame.

O PSO pode se concentrar em convergência ou diversidade em qualquer iteração. Concentrar-se na diversidade significa que as partículas estão dispersas, procurando soluções em uma área grosseiramente grande. Concentrar-se em convergência significa que as partículas estão próximas umas das outras, em busca intensiva de uma pequena área. Uma estratégia promissora é forçar o PSO a concentrar-se em diversidade em iterações iniciais e em convergência nas iterações posteriores (Raquel e Naval, 2005). A Figura 1 ilustra a atualização da posição de uma partícula em um espaço bidimensional. A Figura 2 ilustra o fluxo de funcionamento do PSO.



$v_i(t)$	—	velocidade atual
$x_i(t)$	—	partícula atual
$x_i^{pbest}(t)$	—	melhor solução até o momento
$x_i^{nbest}(t)$	—	melhor solução da vizinhança
$x_i(t+1)$	—	nova solução

Figura 1. Ilustração da atualização da posição de uma partícula x_i em um espaço bidimensional

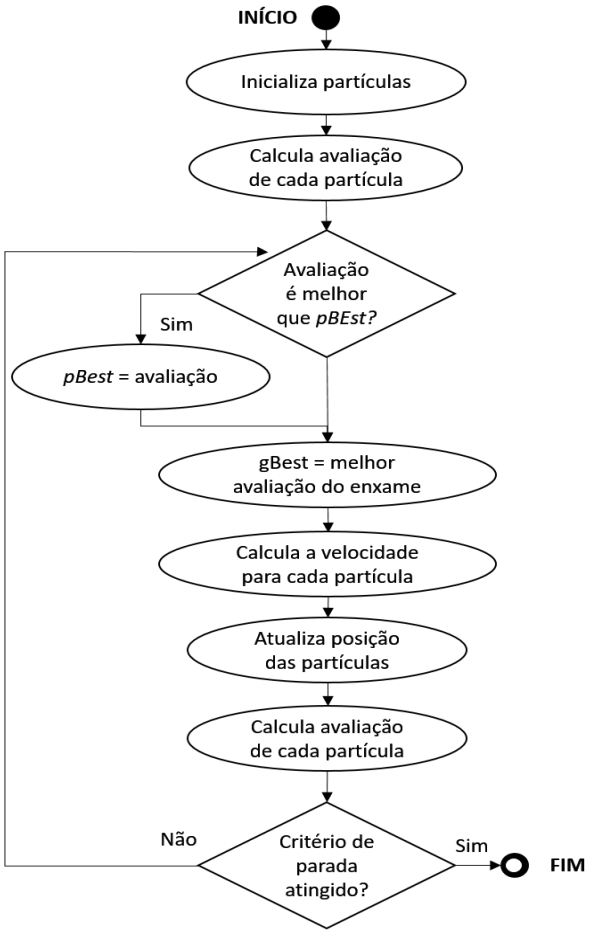


Figura 2. Fluxo do PSO

2.3.1.2. PSO por Melhor Global *(Global Best Particle Swarm Optimization – GBPSO)*

Um algoritmo PSO conhecido como GBPSO apresenta taxas de convergência muito rápidas (Engelbrecht, 2007). Para o GBPSO, a vizinhança de cada partícula é todo o enxame. Portanto, a informação social é a melhor posição encontrada pelo enxame, referido como *gbest*. O grande problema desta abordagem é a grande chance de convergência prematura do algoritmo. Como todo o processo é guiado pela melhor partícula de todo o enxame, o algoritmo pode rapidamente convergir para um ótimo local. A velocidade da partícula *i* é calculada como:

$$v_i(t+1) = v_i(t) + c_1 * rand_1 * (pbest_i(t) - x_i(t)) + c_2 * rand_2 * (gbest(t) - x_i(t)) \quad (5)$$

onde v_i é a velocidade da partícula, c_1 *e* c_2 são constantes positivas de aceleração usadas para dimensionar a contribuição dos componentes cognitivos e sociais, respectivamente, $rand_1$ e $rand_2$ são valores aleatórios no intervalo [0,1] amostrados de uma distribuição uniforme contínua, $pbest$ denota a melhor posição da partícula *i* e $gbest$ denota a melhor posição global do enxame.

2.3.1.3. PSO com Inércia

Há um ponto fraco na atualização das partículas no PSO clássico que pode ser resolvido por meio da introdução de inércia na equação de atualização da velocidade. Como a melhor posição da partícula (*pbest*) e a melhor posição do enxame (*gbest*) melhoram com o tempo, a atualização da velocidade deve se concentrar mais em valores recentes. Esse problema é remediado introduzindo-se na equação de velocidade (Equação (6)) o fator de inércia *w* (Eberhart e Shi, 2000; Shi e Eberhart, 1998), onde $w \in [0,1]$. A nova equação de velocidade da partícula *i* é apresentada abaixo:

$$v_i(t+1) = w * v_i(t) + c_1 * rand_1 * (pbest_i(t) - x_i(t)) + c_2 * rand_2 * (gbest(t) - x_i(t)) \quad (6)$$

onde v_i é a velocidade da partícula, w é o peso da inércia, c_1 *e* c_2 são constantes positivas de aceleração usados para dimensionar a contribuição dos componentes cognitivos e sociais, respectivamente, $rand_1$ e $rand_2$ são valores aleatórios no

intervalo [0,1] amostrados de uma distribuição uniforme contínua, $pbest$ denota a melhor posição da partícula *i* e $gbest$ denota a melhor posição global do enxame.

2.3.1.4. PSO por Melhor Local *(Local Best Particle Swarm Optimization – LBPSO)*

O LBPSO é um caso particular do GBPSO e a diferença entre esse algoritmo e o algoritmo por melhor global é que a equação de velocidade no LBPSO é atualizada utilizando o melhor da vizinhança, ao invés do melhor global. O LBPSO normalmente tem um desempenho melhor, já que evita o problema de convergência prematura, apesar de que em problemas que necessitam de solução em tempo real, uma solução rápida, mesmo que imperfeita, é mais desejada, tornando o GBPSO um algoritmo mais adequado. A velocidade da partícula *i* é calculada como:

$$v_i(t+1) = w * v_i(t) + c_1 * rand_1 * (pbest_i(t) - x_i(t)) + c_2 * rand_2 * (nbest(t) - x_i(t)) \quad (7)$$

onde $nbest$ denota a melhor posição da vizinhança da partícula *i*.

2.3.1.5. PSO por Peso Decrescente (*Decreasing Weight Particle Swarm Optimization – DWPSO)*

O DWPSO é semelhante ao GBPSO, mas o peso de inércia diminui linearmente ao longo do tempo (Engelbrecht, 2007). O objetivo é concentrar-se na diversidade nas iterações iniciais e na convergência nas iterações finais:

$$v_i(t+1) = w(t) * v_i(t) + c_1 * rand_1 * (pbest_i(t) - x_i(t)) + c_2 * rand_2 * (gbest_i(t) - x_i(t)) \quad (8)$$

onde a inércia *w(t)* é calculada a cada iteração segundo a equação:

$$w(t) = w_i - (w_i - w_f)\frac{t}{N} \quad (9)$$

em que *w(t)* é a inércia na iteração *t*, w_i é a inércia designada para a primeira iteração e, w_f é a inércia designada para a última iteração *N*.

O peso da inércia *w* a cada iteração *t* depende do número total de iterações *N*. A inércia pode ser interpretada como a fluidez do meio em que a partícula se move. Isso explica porque experimentos mostram que o melhor desempenho é obtido

quando se inicia a inércia com um valor relativamente alto (e.g., 0.9), o que corresponde a um sistema onde as partículas se movem em um meio pouco viscoso e que realiza a exploração mais global. Em seguida, a inércia é gradualmente reduzida para um valor mais baixo (e.g. 0.4), tornando o sistema mais viscoso e beneficiando a exploração local. Por conseguinte, a alteração no número total de iterações irá alterar o comportamento do algoritmo em cada iteração.

2.3.1.6. PSO com Coeficientes de Aceleração Variantes no Tempo (*Time-Varying Acceleration Coefficients Particle Swarm Optimization – TVACPSO)*

O TVACPSO não só muda o peso da inércia *w*, como também os coeficientes de aceleração, isto é, o peso do melhor pessoal c_1 e o peso do melhor global c_2 ao longo do tempo (Engelbrecht, 2007). O objetivo é ter uma alta diversidade em iterações iniciais e uma alta convergência em iterações finais. O peso de inércia *w* é alterado como em DWPSO e a velocidade é dada por:

$$v_i(t+1) = w(t) * v_i(t) + c_1(t) * rand_1 * (pbest_i(t) - x_i(t)) + c_2(t) * rand_2 * (gbest_i(t) - x_i(t)) \quad (10)$$

onde os pesos do melhor pessoal c_1 e do melhor global c_2, a cada iteração *t*, são calculados usando as seguintes equações:

$$c_1(t) = c_{1i} - (c_{1i} - c_{1f})\frac{t}{N}$$
$$c_2(t) = c_{2i} - (c_{2i} - c_{2f})\frac{t}{N} \quad (11)$$

onde $c_1(t)$ é o peso do melhor pessoal na iteração *t*, $c_2(t)$ é o peso do melhor global na iteração *t*, c_{1i} é o peso no melhor pessoal na primeira iteração, c_{1f} é o peso do melhor pessoal na última iteração *N*, c_{2i} é o peso no melhor global na primeira iteração e c_{2f} é o peso do melhor global na última iteração *N*.

2.3.1.7. PSO com Convergência Garantida (*Guaranteed Convergence Particle Swarm Optimization – GCPSO)*

O GCPSO garante que a busca de partículas seja feita dentro de um raio adaptado dinamicamente a cada passo (Bergh, Van den e Engelbrecht, 2002). A técnica aborda o problema da estagnação e aumenta a convergência local usando a

melhor partícula global para procurar soluções aleatoriamente em um raio adaptativo a cada iteração. GCPSO, tal como descrito no trabalho acima, usa a seguinte equação para determinar as velocidades:

$$v_{ig}(t+1) = -x_i(t) + gbest(t) + w(t) * v_{ig}(t) + \rho(t) * (1 - 2 * rand) \quad (12)$$

onde *ig* é o índice da partícula que atualizou o ótimo global mais recentemente e $w(t)$ é o peso da inércia na iteração *t*. $x_i(t)$ denota a melhor posição da partícula *i* no espaço de busca no passo *t* e $gbest(t)$ denota a melhor posição do enxame no passo *t*. Logo, a expressão $gbest(t) - x_i(t)$ é usada para direcionar a posição da partícula *i* para a melhor posição global. A variável $rand$ é um número aleatório no intervalo [0,1] amostrado de uma distribuição uniforme. O raio de busca é controlado pelo parâmetro ρ, que é calculado da seguinte forma:

$$\rho(t+1) = \begin{cases} 2\rho(t), se\ s(t+1) > s_c \\ \frac{1}{2}\rho(t), se\ a(t+1) > a_c \\ \rho(t), caso\ contrário \end{cases} \quad (13)$$

onde s_c é o limiar de sucesso e a_c é o limiar de falha. Sucesso significa que a equação de cálculo de velocidade resultou em uma melhor posição do que o ótimo global anterior. $\rho(0)$ é definido empiricamente como 1, $s_c = 15$ e $a_c = 5$. O número de sucessos e falhas consecutivas é calculado da seguinte forma:

$$s(t+1) = \begin{cases} 0, se\ a(t+1) > a(t) \\ s(t) + 1, caso\ contrário \end{cases}$$

$$a(t+1) = \begin{cases} 0, se\ s(t+1) > s(t) \\ a(t) + 1, caso\ contrário \end{cases} \quad (14)$$

Desta forma, o raio de busca é dobrado após s_c sucessos sequenciais em alcançar uma solução melhor para o problema; e o raio é reduzido pela metade após a_c falhas em se encontrar uma solução melhor. Todas as outras partículas utilizam a mesma equação de velocidade descrita no GCPSO.

2.3.2. Topologia do PSO

Topologia ou vizinhança do PSO refere-se ao agrupamento de partículas em subgrupos. Uma partícula pode se comunicar e trocar informações sobre o espaço de busca apenas com outras partículas na sua vizinhança (Clerc, 2006). O desempenho do algoritmo PSO depende em certa medida da topologia de

vizinhança. As partículas formam um grupo, e uma partícula *j* situa-se na vizinhança da partícula *i* se houver um elo da partícula *i* para a partícula *j*. Isto significa que a partícula *j* informa à partícula *i* sobre a sua posição no espaço de busca. A partícula *j* é chamada de partícula informante, enquanto que a partícula *i* é chamada de partícula informada (Clerc, 2006). Cada partícula é um membro da sua vizinhança. O tamanho da vizinhança refere-se ao número de partículas na vizinhança.

2.3.2.1. Topologia de Vizinhança Estática

A topologia da vizinhança é definida pela chamada matriz de adjacência m_{ij}, em que as linhas correspondem às partículas informantes e as colunas, às partículas informadas. Em geral, a matriz é não simétrica e contém zeros e uns como entradas. A matriz tem sempre uns na diagonal, porque cada partícula está contida na sua própria vizinhança. Usando a matriz de adjacência, é possível definir diferentes tipos de topologias de vizinhança. A Figura 3 ilustra topologias diversas para um enxame com oito partículas.

A topologia em estrela mostrada na Figura 3(a) tem apenas uma vizinhança e cada partícula tem um elo para todas as outras partículas. Algoritmos PSO que fazem uso desta topologia são chamados de *Global Best* ou algoritmos *gbest* (Engelbrecht, 2005). A utilização desta topologia conduz a uma convergência rápida, embora o algoritmo seja suscetível a ficar estagnado em mínimos locais. A matriz de adjacência para a topologia em estrela é uma matriz com uns para todas as entradas.

As topologias da Figura 3(b) e Figura 3(c) têm mais do que uma vizinhança. Na topologia em anel (Figura 3(b)), cada partícula tem uma ligação para duas partículas adjacentes; assim, cada vizinhança contém um total de três partículas. As vizinhanças na estrutura do anel são sobrepostas, porque cada partícula reside simultaneamente em três vizinhanças. Nesta topologia, partes da população que estão distantes entre si são também independentes umas das outras. Então, um segmento da população pode convergir para um mínimo local, enquanto outra parte converge para outra solução ou continua a busca. A influência se espalha de vizinhança a vizinhança até que um ótimo realmente se destaque como um melhor

encontrado pela população como um todo, e, eventualmente, atraia todas as partículas.

Na topologia de agrupamento, as partículas podem se organizar em grupos e o conceito de vizinhança é aplicado individualmente a cada grupo. No exemplo da Figura 3(c), as oito partículas são colocadas em duas vizinhanças, cada uma contendo quatro partículas. Algoritmos PSO usando as topologias em anel e em agrupamento são chamados de *local best* ou *lbest* (Engelbrecht, 2005). As topologias de vizinhança podem ser fixas ou podem ser variadas ao longo das iterações.

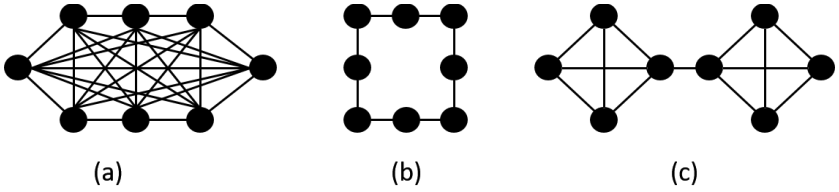


Figura 3. Exemplos de topologias de vizinhança para um sistema de oito partículas. (a) topologia em estrela; (b) anel; (c) agrupamento aleatório.

Na topologia em roda, ilustrada na Figura 4(a), as partículas são efetivamente isoladas umas das outras, e toda a informação deve ser comunicada a uma partícula central. Essa partícula compara o desempenho de todas as outras partículas do enxame, e ajusta a sua trajetória baseado na melhor solução. Se as mudanças resultarem em uma melhora na partícula central, então essa melhora é comunicada para o restante da população. Assim, a partícula central serve como um amortecedor, diminuindo a velocidade de transmissão de boas soluções pelo restante do enxame.

Na topologia Von Neumann (Kennedy e Mendes, 2002), apresentada na Figura 4(b), é formada uma estrutura em malha, onde cada indivíduo pode se comunicar com até quatro vizinhos.

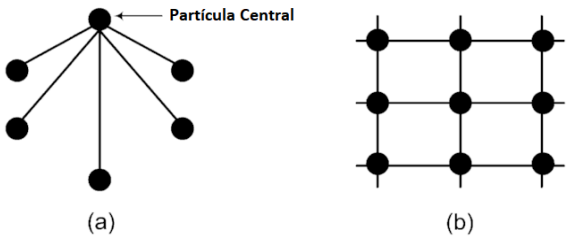


Figura 4. Topologias em roda (a) e Von Neumann (b)

2.3.2.2. Topologia de Vizinhança Dinâmica

Na topologia de vizinhança variável cada partícula tem elos para somente algumas partículas do enxame, que são alterados dinamicamente.

Em (Suganthan, 1999), em função de a topologia *lbest* ser melhor para explorar o espaço de busca, enquanto a *gbest* converge mais rápido, optou-se por começar a pesquisa com uma rede em anel e aumentar lentamente o tamanho da vizinhança até que a população estivesse totalmente conectada ao final do ciclo de otimização. Esse trabalho também relatou resultados em outros tipos de topologia, em que os vizinhos foram definidos pela proximidade no espaço de busca e o número de vizinhos aumentou dinamicamente ao longo da otimização.

Em (Liang *et al.*, 2012), os autores criaram subpopulações aleatórias de tamanho *n* e geraram todas as conexões aleatoriamente.

(Janson e Middendorf, 2005) organizaram as partículas em uma hierarquia dinâmica, com cada partícula sendo influenciada pelo seu próprio sucesso e pelo da partícula diretamente acima dela. As partículas com melhor desempenho são movidas para cima na hierarquia. Essa disposição teve mais efeito sobre as partículas com piores avaliações. O resultado foi um melhor desempenho na maioria dos *benchmarks* considerados.

Basicamente, a topologia dinâmica utiliza algum método para atualizar as vizinhanças. Esta abordagem é robusta e reduz a suscetibilidade de soluções ficarem estagnadas em um ótimo local (Poli, Kennedy e Blackwell, 2007).

2.4. Otimização com Restrições

Os tipos de manipulação de restrições em problemas de otimização desenvolvidos ao longo dos anos podem ser, grosseiramente, classificados em três categorias (Coello Coello e A., 2010; Kramer e Oliver, 2010):

- Penalização;
- Viabilidade de soluções;
- Multiobjetivo.

2.4.1. Penalização

Os problemas de otimização com restrições são abundantes e a abordagem mais comum em algoritmos evolutivos para lidar com restrições é o uso de penalidades. A ideia principal da penalização é transformar problemas de otimização com restrições em problemas sem restrições adicionando (ou subtraindo) um certo valor na função objetivo com base na quantidade de violação das restrições presente em uma determinada solução. Assim, os métodos de penalização permitem a resolução de problemas com restrições por métodos tipicamente utilizados em problemas sem restrições. As técnicas de penalização podem ser classificadas como multiplicativas ou aditivas. No caso das multiplicativas, um fator de penalização positivo maior do que 1 é utilizado com o intuito de aumentar o valor da avaliação dos indivíduos inválidos em problemas de minimização. No caso das aditivas, uma função de penalização é adicionada à função objetivo dos indivíduos inválidos. O método mais usado para as penalizações aditivas é definido pela equação 15, tal que p(x) = 0 se x obedecer todas as restrições do problema.

$$F(x) = f(x) + p(x) \qquad (15)$$

onde *F(x)* é a função objetivo expandida a ser minimizada, *f(x)* é a função objetivo original e *p(x)* é o valor da penalização.

As técnicas de penalização aditivas podem ser divididas em dois tipos: exterior e interior. No caso das técnicas exteriores, inicia-se em uma região inválida e move-se em direção a uma região válida. Nas técnicas interiores, o termo de penalização é escolhido de modo que o seu valor seja pequeno em pontos distantes da fronteira e maiores em pontos perto da fronteira. Ainda que a implementação de uma função de penalização seja considerada bastante simples, essas funções são altamente dependentes do problema e exigem um ajuste cuidadoso de seus fatores de penalização, a fim de determinar o rigor das penalizações a serem aplicadas. Uma das dificuldades dos métodos de penalização é encontrar parâmetros de penalidade convenientes de forma que a solução *x* corresponda, de alguma forma, ao mínimo (máximo) do respectivo problema com restrições. As técnicas de manipulação de restrições baseadas em funções de penalização mostram muita

diversidade na definição dos fatores de penalização. As penalizações podem ser: estática, dinâmica e adaptativa.

(i) *Penalização Estática:* as penalidades permanecem constantes durante todo o processo evolutivo. A função objetivo penalizada é, então, a função objetivo da solução candidata mais uma "multa" (para problemas de minimização). Pode ocorrer de se penalizar uma determinada solução candidata com violações pequenas da mesma forma que se penaliza uma outra solução com violações maiores. Isto pode levar uma partícula próxima ao ótimo a se afastar, dificultando a busca da solução desejada.

(ii) *Penalização Dinâmica* (Jan, Tairan e Khanum, 2013): surgiram como alternativa de suprir a dificuldade do usuário de determinar os coeficientes de penalização – uma desvantagem das penalizações estáticas. Nas penalizações dinâmicas, o período ou instante do processo evolutivo está diretamente envolvido com os coeficientes de penalização.

(iii) *Penalização Adaptativa:* nas funções com penalização adaptativa, os parâmetros de penalização são atualizados ao longo do processo de evolução, de acordo com informações coletadas da população. Assim, eles podem ser alterados com base nas gerações, no grau de violação das restrições do problema, na função objetivo, entre outros.

(iv) *Penalização competitiva* (Deb, 2000; Yu *et al.*, 2014): é usada uma função de penalização para soluções inválidas e um operador de competição para escolher a melhor entre um par de soluções: (a) entre duas soluções válidas, a melhor será a que obtiver melhor resposta da função de avaliação; (b) entre uma solução válida e uma inválida, a solução válida é considerada melhor; (c) entre duas soluções inválidas, é escolhida a que tiver violado menos restrições.

(v) *Death Penalty* (Kramer e Schwefel, 2006): nessa abordagem, soluções inválidas são rejeitadas e novas soluções são geradas até que toda uma população válida seja criada.

2.4.2.
Viabilidade de Soluções

Os operadores de preservação de validade obrigam as soluções candidatas a serem válidas. Um exemplo conhecido é o algoritmo GenocopIII (Michalewicz e Nazhiyath, 1995), que se baseia em conceitos de co-evolução e algoritmos de reparação de solução. Uma abordagem de predador-presa para lidar com restrições é proposta por Paredis (Paredis, 1994) usando duas populações separadas. Schoenauer e Michalewicz (Schoenauer e Michalewicz, 1996) propõem operadores especiais que são projetados para pesquisar regiões na proximidade de restrições ativas. Uma visão abrangente das técnicas de manipulação de restrições baseadas em decodificadores é dada por Coello (Coello Coello e A., 2010).

Os algoritmos de reparo substituem soluções inválidas ou apenas usam as soluções reparadas para avaliação de seus pendentes inválidos (Coello, 2002). Estes algoritmos também podem ser vistos como métodos de busca locais que reduzem a violação de restrições. O algoritmo de reparo gera uma solução válida a partir de uma inválida. No caso Baldwiniano (Whitley, Gordon e Mathias, 1994), a aptidão da solução reparada substitui a aptidão da solução original. No caso Lamarckiano (Whitley, Gordon e Mathias, 1994), a solução válida substitui a inválida. Em geral, definir um algoritmo de reparo pode ser tão complexo quanto resolver o problema em si.

2.4.3.
Multiobjetivo

As técnicas de otimização multiobjetivo baseiam-se na ideia de lidar com cada restrição como um objetivo. Sob este pressuposto, muitos métodos de otimização multiobjetivo podem ser aplicados. Essas abordagens foram utilizadas por Yu et al. (Yu *et al.*, 2014) e Li et al. (Li *et al.*, 2016). No método de memória comportamental por Schoenauer e Xanthakis (Schoenauer e Xanthakis, 1993), o algoritmo evolucionário concentra-se em minimizar a violação de restrição de cada restrição em uma determinada ordem e otimizar a função objetivo na última etapa.

3.
Modelo Proposto

Como mostrado na Figura 5, o algoritmo PSO+ proposto para lidar com restrições lineares e não lineares compreende seis módulos principais (Kohler *et al.*, 2016):

- Tratamento de equações em inequações – conversão de restrições de igualdade em restrições de desigualdade;
- Heurística de inicialização do enxame de partículas válido – heurística que garanta a geração de indivíduos válidos para enxame inicial do algoritmo;
- Pontos de apoio – pontos totalmente inválidos que ajudam a busca por soluções ótimas na borda do espaço de busca válido e insiram diversidade no enxame;
- Múltiplos enxames – dois enxames que desempenham papéis diferentes no processo de otimização;
- Operador aritmético entre partículas – operador que repara soluções inválidas e garante a geração de partículas válidas durante a evolução;
- Topologia de vizinhança – topologia que determina a forma como a informação é passada para outras partículas do enxame.

Figura 5. Visão Macro do Algoritmo PSO+

Cada módulo do algoritmo é descrito em detalhes nas seções que seguem.

3.1. Módulo de Tratamento de Restrições Lineares e Não Lineares de Igualdade

As restrições de igualdade devem ser convertidas em restrições de desigualdade – abordagem comum ao se tratar problemas de otimização com restrições de igualdade. Supondo uma restrição de igualdade:

$$f(x_1, x_2, \dots, x_n) = b \qquad (16)$$

A equação de restrição será substituída pela seguinte restrição de desigualdade:

$$|f(x_1, x_2, \dots, x_n) - b| \leq \varepsilon \qquad (17)$$

onde ε é a tolerância permitida (um valor bem pequeno), *f(x)* é qualquer equação linear ou não linear. Neste trabalho, $\varepsilon = 10e - 7$.

3.2. Módulo de Inicialização do Enxame

Em algoritmos estocásticos baseados em populações como os evolucionários (EAs, do inglês *Evolutionary Algorithms*), a qualidade da população inicial gerada aleatoriamente desempenha um papel importante em seu desempenho. Se a população inicial possuir algumas boas soluções, os algoritmos convergem

rapidamente. No entanto, não se espera que as soluções aleatórias sejam sempre de boa qualidade. Alguns algoritmos como Genocop (Michalewicz e Janikow, 1996), o método de mapeamento homomorfo de Koziel e Michalewicz (Koziel, S., Michalewicz, 1999) e o algoritmo proposto nesse trabalho supõem um ponto de partida válido, o que significa que o usuário ou o algoritmo devam ter uma maneira de gerá-lo em um tempo razoável (Coello, 2002).

Dois métodos de inicialização da população são propostos. O primeiro é utilizado para inicialização da população para os problemas *benchmark* descritos nas seções 4.1 e 4.2, e o segundo é empregado no caso real de otimização de alocação de poços descrito na seção 4.3.

O segundo método de inicialização de enxames é mais complexo e muito dependente do problema e de suas restrições. Por este motivo, ele somente é utilizado neste trabalho para iniciar o enxame do estudo de caso real. Como será explicado na seção 4.3.4, é necessário gerar uma função de distribuição para cada uma das restrições, o que tornaria o processo muito custoso se realizado para os 29 *benchmarks*, onde, para cada *benchmark*, o número de restrições varia de 1 a 729.

3.2.1. Heurística de Inicialização da População

Como alguns problemas podem ter restrições de equações ou inequações lineares ou não lineares que tornam difícil a inicialização da população com partículas válidas, uma heurística é proposta, como descrito na Figura 6 e na Figura 7. Os três passos são detalhados abaixo:

1. Uma primeira partícula é sorteada aleatoriamente até que se torne válida. Em seguida, o resto do enxame é iniciado. Se a próxima partícula é válida, ela será salva; se não o for, uma partícula totalmente válida iniciada anteriormente é utilizada para trazer a partícula inválida para o espaço de busca válido até que se torne válida;
2. Se ainda assim não for possível gerar a população inicial em tempo hábil, o processo de inicialização chega a um segundo passo, onde é feita uma otimização com PSO (por melhor global) com o objetivo de minimizar o número de restrições violadas;

3. Um terceiro passo foi implementado para o caso dos dois passos anteriores não terem sido capazes de gerar o enxame válido, baseado no algoritmo SSRT (do inglês *Search Space Reduction Technique*) de Ullah et al. (Ullah *et al.*, 2008). Esse passo recebe a população do passo anterior (minimização do número de violações de restrições) e começa um processo que permite que os agentes mais inválidos se movam para a região válida, "espremendo" o espaço de busca. Isso significa que o processo evolutivo começa com uma população melhor em um espaço de busca reduzido.

Algoritmo Heurística de Inicialização da População: Passo 1

```
1:  Faça
2:  {
3:          Inicie uma partícula aleatoriamente
4:   } Enquanto (tentativas < 5000 e partícula inválida)

5:  Se partícula é válida
6:  {
7:          Salva partícula no enxame de partículas válidas
8:  }
9:  Senão
10: {
11:         Passe para o Passo 2
12: }
13: Faça
14: {
15:     Aleatoriamente inicia uma partícula X (válida ou inválida)
16:     Se partícula X é válida
17:     {
18:         Salva no enxame de partículas válidas
19:         Continue (Tente iniciar uma próxima partícula)
20:     }
21:     Senão
22:     {
23:         Aleatoriamente escolha uma partícula F válida
24:         Faça
25:         {
26:             Nova partícula X = a*(X) + (1-a)*F  //a ∈ [0,1]
27:         } Até que a nova partícula X seja válida

28:         Salve a nova partícula X no enxame de partículas válidas
29:     }
30: } Enquanto (tentativas < 5000 ou enxame totalmente válido)
```

Figura 6. Algoritmo da heurística de inicialização da população: Passo 1

Algoritmo Heurística de Inicialização da População: Passos 2 e 3

```
1:  //PASSO 2:
2:  Otimização com PSO onde objetivo é minimizar o número de restrições violadas.
3:  Verifique a validade das partículas.
    Se conseguiu gerar alguma partícula válida, adicione-a ao enxame de partículas válidas.

4:  //PASSO 3:
5:  Crie uma lista de partículas inválidas retornadas da otimização do Passo 2 e ordene-a, em ordem crescente, pelo número de restrições violadas.

6:  Calcule a diferença líquida entre a saída esperada do limite de cada restrição e a saída da restrição com os valores da partícula inválida (Violação Líquida Total de todas as restrições)

7:  Marque as restrições que foram violadas

8:  Ordene as soluções inválidas baseadas na Violação Líquida Total

9:  Faça
10: {
11:     Calcule o centroide entre a partícula inválida e 10% das melhores partículas (k-means com distância euclidiana)
12:     Salve a partícula-centroide se for válida, e passe para próxima inválida

13:     Mova 10% das partículas inválidas (incluindo a partícula que está sendo analisada) em direção ao centroide
14:     Salve a partícula se for válida, e passe para próxima partícula inválida

15:     Selecione a restrição, dentre as violadas pela partícula, que tem a maior violação líquida total

16:     Selecione, aleatoriamente, uma variável que esteja envolvida na restrição violada e que ainda não tenha sido modificada

17:     Gere δ aleatoriamente com distribuição gaussiana (média = 0, desvio padrão = 1)
18:     Modifique a variável em ±δ e marque a variável como modificada

19:     Se partícula se tornar válida
20:     {
21:         Continue (próxima iteração)
22:     }
23:     Se todas as restrições estão marcadas e todas as variáveis foram modificadas
24:     {
25:         Calcule o centroide entre a partícula inválida e 10% das melhores partículas, juntamente com a partícula modificada nessa iteração
26:     }
27: } Enquanto partícula for inválida
```

Figura 7. Algoritmo da heurística de inicialização da população: Passos 2 e 3

O passo 3 guia a população inicial em direção à região válida. As partículas válidas (ou melhor avaliadas no passo 2), juntamente com a partícula inválida a ser validada, são utilizadas para gerar um centroide. Uma porcentagem das partículas inválidas é encorajada a se mover em direção ao centroide. Ao aplicar o SSRT, as partículas geradas aleatoriamente deixam de ser aleatórias e começam a aprender a direção do espaço de busca válido, o que ajuda a inicialização mais rápida das partículas válidas e melhora a qualidade da solução.

O valor de δ é bem pequeno – δ = |G(0,1)|, onde G(0,1) é um gerador aleatório com distribuição gaussiana com média zero e desvio padrão 1.

Para calcular o centroide, para cada variável do centroide x_i^c, foi usado o algoritmo *k-means* (distância euclidiana), e as partículas inválidas que seguem o centroide o fazem de acordo com a seguinte equação:

$$x_i^n = \alpha x_i + (1-\alpha)x_i^c,\ i = 1, \dots, n; \qquad (18)$$

onde x_i^n e x_i^c são a i-ésima partícula a ser movida e o centroide, respectivamente; *n* é o número de variáveis no vetor solução e α é um número aleatório gerado com distribuição uniforme no intervalo [0,1].

3.2.2. Metropolis-Hastings

Inicialmente proposta para estatística e física estatística, o algoritmo Metropolis-Hastings é um método de Cadeia de Markov Monte Carlo (MCMC, do inglês *Markov Chain Monte Carlo*) para a obtenção de uma sequência de amostras aleatórias de uma distribuição de probabilidade cuja amostragem direta é difícil (Hastings, 1970). Esta sequência pode ser usada para aproximar a distribuição ou para calcular uma integral. O Metropolis-Hastings e outros algoritmos MCMC são geralmente usados para a amostragem de distribuições multidimensionais, especialmente quando o número de dimensões é alto.

O algoritmo Metropolis-Hastings é uma ferramenta de simulação estocástica que propõe uma forma genérica para construir uma cadeia de MCMC no espaço de estado *X*, o qual é ergódico (todos os estados devem ser recorrentes e aperiódicos) e estacionário em relação a uma distribuição π, ou seja:

Se $X(t) \sim \pi(x)$, então $X(t+1) \sim \pi(x)$ e que, portanto, converge para π (Firmino, 2009).

Assim, a cadeia de Markov $X(1), X(2), .., X(t)$ retornada pelo método é tal que $X(t)$ está convergindo para π, significando que a Cadeia de Markov pode ser considerada como uma amostra, embora uma amostra dependente, e aproximadamente distribuída de π.

O algoritmo de Metropolis-Hastings, dada uma densidade de probabilidade π chamada alvo (*target*) e definida num espaço X, funciona da seguinte maneira:

- Escolhe-se uma densidade condicional *Q* (denominada proposta ou *kernel* do candidato);
- A transição do valor da cadeia de Markov (*X (t)*) no tempo *t* para o valor no momento *t + 1* procede de acordo com o passo de transição definido no pseudocódigo mostrado na Figura 8.

Pseudocódigo do Algoritmo Metropolis-Hastings

1: Dado $\boldsymbol{X_t = x_0}$,
2: **Repetir para** $\boldsymbol{t = 1, 2, 3 \ldots N}$
3: {
4: **Gerar** $\boldsymbol{Y_t \sim Q(y|x_t)}$
5: **Calcular** $\boldsymbol{a = a1 * a2}$
Onde
$\boldsymbol{a1 = \frac{\pi(y)}{\pi(x_t)}}$ (razão de probabilidade entre amostra proposta y e a amostra anterior x_t)
$\boldsymbol{a2 = \frac{Q(x_t|y)}{Q(y|x_t)}}$ (razão da densidade proposta: de x_t para y e vice-versa).
6: **O novo estado x_{t+1} é escolhido de acordo com:**
Se $\boldsymbol{a \geq 1}$:
$\boldsymbol{X_{t+1} = Y_t}$,
Caso contrário
$\boldsymbol{X_{t+1} = \begin{cases} Y_t & com\ probabilidade\ a \\ x_t & com\ probabilidade\ 1 - a \end{cases}}$,
7: **}**
8: **Retorna os valores {X(1), X(2), . . , X(N)}**

Figura 8. Pseudocódigo do algoritmo Metropolis-Hastings

O uso da probabilidade de aceitação a é fundamental para garantir que a cadeia construída seja ergódica, e portanto, se obtenha uma distribuição estacionária. Desta forma, se o ponto candidato Y for aceito, o próximo estado será $X_{t+1} = Y$; se o candidato for rejeitado, a cadeia não se moverá, i.e. $X_{t+1} = X_t$.

Mais detalhes da implementação desse algoritmo são fornecidos na seção 4.3.4, para um caso real de otimização.

3.3. Módulo de Pontos de Apoio

Uma das principais razões que torna difícil a localização da solução ótima global é que os sistemas evolucionários não têm a capacidade de pesquisar precisamente a área de fronteira entre regiões válidas e inválidas do espaço de busca. Esta capacidade é bastante importante no caso de problemas de otimização com restrições de igualdade não lineares ou com restrições não lineares ativas no ótimo (Michalewicz e Schoenauer, 1996). O número de restrições ativas no ótimo é, naturalmente, importante: se mais restrições estão ativas no ótimo, algoritmos que realizam buscas perto da fronteira da região válida são mais propensos a ter sucesso.

Técnicas de computação evolucionária têm um enorme potencial para a incorporação de operadores especializados que buscam a fronteira das regiões válidas e inválidas de uma forma eficiente. Como em muitos problemas de otimização com restrição, algumas restrições estão ativas no ótimo global, logo este se encontra no limite do espaço válido. Já que restringir o tamanho do espaço de busca em algoritmos evolucionários é geralmente benéfico, parece natural no contexto da otimização com restrições restringir a procura de uma solução próxima ao contorno da parte válida do espaço.

Um novo conceito é introduzido neste trabalho para melhorar a convergência do PSO. Pontos de apoio são pontos inválidos usados para redirecionar as partículas com uma determinada probabilidade. Eles auxiliam o algoritmo a encontrar soluções que estejam na fronteira do espaço de busca válido, tal como ilustrado na Figura 9. Na Figura 10 é apresentado o pseudocódigo deste módulo. Eles também mantêm a diversidade do enxame, enquanto mudam a tendência das partículas de se direcionarem no sentido da melhor partícula do enxame.

Pontos de apoio diferem das partículas pois eles não se movem no espaço de busca – em vez disso, eles são trocados de tempos em tempos (o intervalo de tempo é previamente definido) –, não são avaliados e não pertencem a nenhum enxame. Os pontos de apoio são iniciados aleatoriamente e não devem pertencer ao espaço de busca válido. Cada vez que uma partícula válida se move no espaço de busca, há uma chance (20% – valor definido empiricamente) de ela ser redirecionada e influenciada por uma posição escolhida aleatoriamente a partir do ponto de apoio

do grupo. As partículas que forem redirecionadas por um ponto de apoio são movidas na direção deste até atingir um critério de parada: momento em que deixam de ser válidas ou um número *x* de movimentações (definido nesse trabalho como dez) é alcançado. As equações que descrevem a busca das partículas pela borda do espaço de busca válido e pelo aumento na diversidade da população em direção ao ponto de apoio são:

$$v_{max} = P - F \quad (19)$$

$$v_i(t+1) = v_i(t) * rand * (F - x_i(t)) \quad (20)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (21)$$

onde v_{max} é a velocidade máxima que a partícula pode atingir, P denota o vetor solução que representa a partícula a ser direcionada ao ponto de apoio e F denota o vetor que indica a localização do ponto de apoio no espaço multidimensional, o vetor $v_i(t)$ é a velocidade da partícula i no instante t, *rand* é um número aleatório no intervalo [0,1] amostrados de uma distribuição uniforme contínua, e x_i é o vetor posição da partícula i.

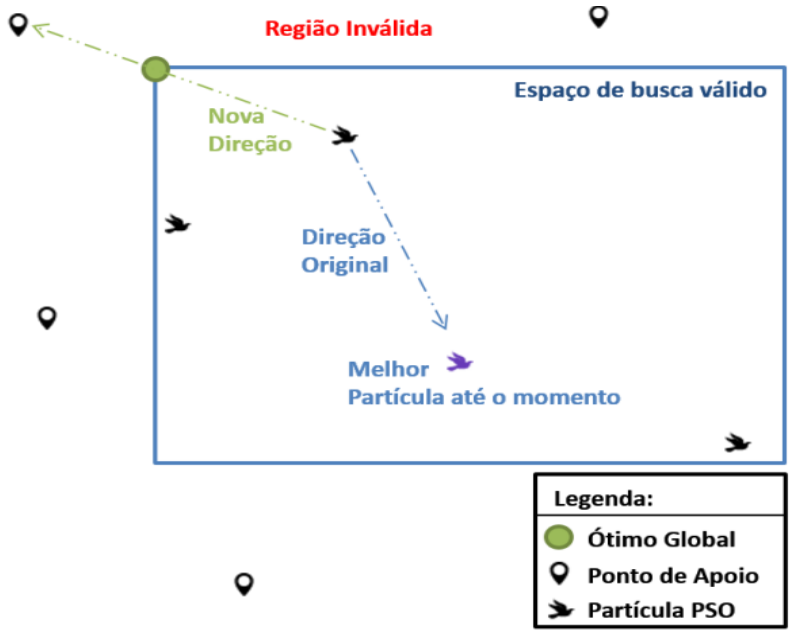


Figura 9. Pontos de Apoio

Pseudocódigo do Funcionamento dos Pontos de Apoio

```
1:  ProbabilidadeRedirecionamento = 0,2
2:  Se válida (P) //partícula P
3:  {
4:      Se rand() <= ProbabilidadeRedirecionamento
5:      {
6:          Enquanto partícula for válida
7:          {
8:              vMax = P – F //F (foothold) = ponto de apoio
9:              v(t+1) = v(t) + * rand() * (F – x(t))
10:             x(t+1) = x(t) + v(t+1)
11:             Se válida (P em x(t+1))
12:             {
13:                 Atualiza posição de P
14:             }
15:             Caso contrário
16:             {
17:                 Sai do Loop
18:             }
19:         }
20:     }
21: }
```

Figura 10. Pseudocódigo do Funcionamento dos Pontos de Apoio

3.4. Módulo de Múltiplos Enxames

Dois enxames separados são criados com o objetivo de separar os papéis de cada partícula e auxiliar na busca de soluções válidas durante o processo de evolução. O primeiro é chamado de enxame de referência e consiste em partículas totalmente válidas, isto é, partículas que satisfazem todas as restrições do problema. O segundo, denominado enxame de fronteira, consiste em partículas que devem, pelo menos, satisfazer as restrições de domínio do problema. O enxame de fronteira atua como um enxame de busca, já que pode ser parcialmente inválido. Esta abordagem baseia-se no GenocopIII (Algoritmo Genético para Otimização Numérica de Problemas com Restrições) (Michalewicz e Nazhiyath, 1995). Os diferentes papéis de cada um dos enxames são importantes quando se tem, por exemplo, um espaço de busca com regiões válidas descontínuas, já que nessa situação, o algoritmo pode ficar preso em ótimo local por não ter conhecimento dos vários conjuntos de regiões válidas.

3.5.
Módulo de Operador Aritmético

As partículas de referência são diretamente avaliadas no início do processo de otimização, assim como as partículas de fronteira válidas. No entanto, as partículas de fronteira inválidas são "reparadas" antes de se realizar a avaliação. Este processo de reparação consiste na seleção de uma partícula de referência (*Ri*) e a aplicação de um operador de recombinação entre a partícula de fronteira inválida (*Fi*) e *Ri* até que uma nova partícula válida (*Z*) seja encontrada e, em seguida, avaliada. Além disso, se a avaliação de *Z* é melhor do que a avaliação de *Ri*, esta partícula substitui *Ri* no enxame de referência. Além disso, Z pode substituir *Fi* com certa probabilidade. A Figura 11 ilustra o procedimento do operador de crossover aritmético, onde os espaços de busca válidos estão dentro das formas. O pseudocódigo do funcionamento do operador é apresentado na Figura 12.

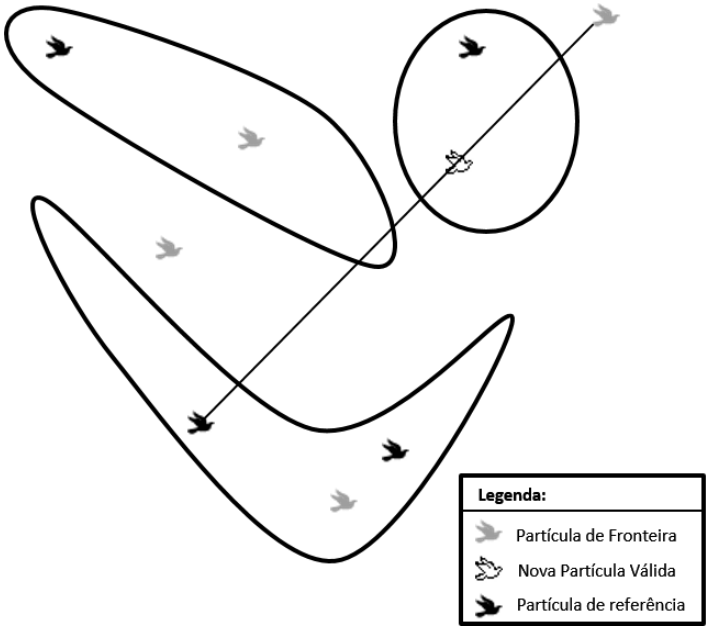


Figura 11. Procedimento de Crossover Aritmético.

Pseudocódigo do Funcionamento do Operador de Cruzamento Aritmético

```
1:  ProbabilidadeSubstituicao = rand([0,1]) //rand() ∈ [0,1]
2:  Se inválida (F)
3:  {
4:      Z = a*F + (1 - a)*R //a ∈ [0,1]
5:      Enquanto inválida (Z)
6:      {
7:          Z = a*Z + (1 - a)*R //a ∈ [0,1]
8:      }
9:      Se avaliação (Z) > avaliação (R)
10:     {
11:         R = Z
12:     }
13:     Se rand() < ProbabilidadeSubstituicao
14:     {
15:         F = Z
16:     }
17: }
```

Figura 12. Pseudocódigo – Operador de Cruzamento Aritmético

3.6. Módulo de Topologia de Vizinhança

Conforme apresentado na seção 2.3.2, o PSO utiliza topologia para especificar como a informação de busca pelo ótimo é "divulgada" para cada partícula do enxame. Definir uma topologia em um algoritmo de otimização por enxame de partículas é especialmente importante para evitar o problema de convergência prematura: se todas as partículas têm informação de todo o enxame a todo momento, rapidamente todas as partículas vão se direcionar para a melhor solução corrente – que pode ser um ótimo local – sem explorar devidamente o espaço de busca. A topologia proposta neste trabalho, denominada aqui *vizinhança de agrupamento aleatório coordenado*, é uma junção de dois tipos de topologia de vizinhança: agrupamentos aleatórios e em anel com indivíduo central – explicados na seção 2.3.2.1. Nesta topologia, são formados grupos aleatórios, onde cada grupo é formado por cinco partículas, sendo uma delas a partícula central. Todos os agrupamentos são isolados entre si e somente se comunicam através das suas respectivas partículas centrais, que compara o desempenho de todas as partículas centrais, e ajusta a sua trajetória baseado na melhor solução. Se as mudanças resultarem em uma melhora na partícula central, então esta melhora é comunicada para o restante do agrupamento. Assim, a partícula central serve como um

amortecedor, diminuindo a velocidade de transmissão de boas soluções pelo restante do enxame. A topologia proposta é ilustrada na Figura 13.

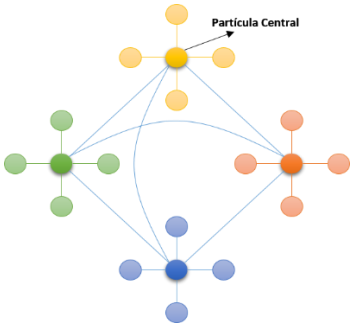


Figura 13. Vizinhança de agrupamento aleatório coordenado

Em função disso, a velocidade da partícula *i* é atualizada segundo a equação 22. A partícula central segue a equação 23 e se houver uma melhora na função objetivo por parte de alguma das partículas de sua vizinhança, as partículas da vizinhança seguem a equação de velocidade 24.

$$v_i(t+1) = w(t) * v_i(t) + c_1 * rand_1 * (pbest_i(t) - x_i(t)) + c_2 * rand_2 * (nbest(t) - x_i(t)) \quad (22)$$

$$v_i(t+1) = w(t) * v_i(t) + c_1 * rand_1 * \left(pbest_i(t) - x_i(t)\right) + c_2 * rand_2 * (gbest(t) - x_i(t)) \quad (23)$$

$$v_i(t+1) = w(t) * v_i(t) + c_1 * rand_1 * \left(pbest_i(t) - x_i(t)\right) + c_2 * rand_2 * (agbest(t) - x_i(t)) \quad (24)$$

onde v_i é a velocidade da partícula, w é o peso da inércia, c_1 *e* c_2 são constantes positivas de aceleração usados para dimensionar a contribuição dos componentes cognitivos e sociais, respectivamente, $rand_1$ *e* $rand_2$ são valores aleatórios no intervalo [0,1] amostrados por uma distribuição uniforme contínua, $pbest$ denota a melhor posição da partícula i e $nbest$ denota a melhor posição da vizinhança enxame (sem a partícula central); $gbest$ denota a melhor posição de todo o enxame, que somente é informada às partículas de um grupo quando as movimentações do mesmo implicam em uma melhora na avaliação da partícula central, e $agbest$, a melhor posição de todo os agrupamento, incluindo a partícula central. O parâmetro de inércia *w* é ajustado dinamicamente com o passar das iterações de forma que a

evolução se concentre na diversidade nas iterações iniciais e na convergência nas iterações finais:

$$w(t) = w_{inicial} - (w_{inicial} - w_{final})\frac{t}{N} \qquad (25)$$

onde *t* é a iteração corrente e *N* é o número total de iterações.

Além disso, a velocidade da partícula é limitada por um valor mínimo e um valor máximo:

$$v_{i_{max}} = ub - x_i$$
$$v_{i_{min}} = lb - x_i \qquad (26)$$

sendo *ub* o vetor que representa as restrições máximas de domínio do problema e *lb* as restrições mínimas. A Figura 14 ilustra o fluxograma do modelo proposto.

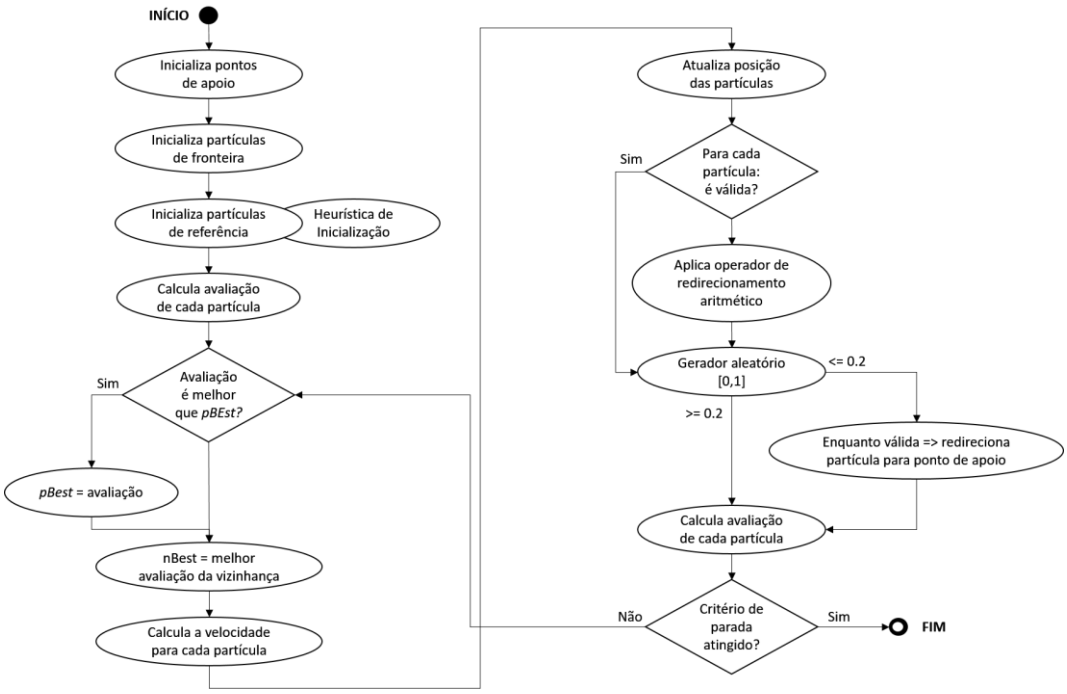


Figura 14. Fluxograma do modelo PSO+

4
Estudos de Caso

Este capítulo apresenta os resultados do modelo proposto PSO+ em três grupos de problemas: (*i*) **Estudo de Caso I**: cinco *benchmarks* foram utilizados para avaliar os principais módulos e o desempenho do PSO+ em problemas de otimização com restrições de domínio. (*ii*) **Estudo de Caso II**: são testados vinte e quatro funções *benchmark*, todas elas criadas artificialmente (Liang *et al.*, 2006) e propostas em uma competição de algoritmos de otimização para avaliar o desempenho do PSO+ perante outras implementações de PSO, assim como outros algoritmos evolucionários. (*iii*) **Estudo de Caso Real**: o PSO+ é avaliado em um problema real de otimização de alocação de poços em um reservatório de petróleo e comparado com resultados obtidos utilizando o GenocopIII (Michalewicz e Nazhiyath, 1995).

4.1.
Estudo de Caso com Funções *Benchmark* para Otimização com Restrições de Domínio

4.1.1.
Benchmarks

Para avaliar separadamente cada módulo implementado no algoritmo, foram utilizadas cinco funções *benchmark*. As condições de cada otimização realizada são apresentadas a seguir:

- Todas as funções foram simuladas com 10 variáveis, isto é, $n = 10$, com os correspondentes domínios definidos a seguir;
- Em cada função foram realizados 10 experimentos independentes, sendo calculados, ao final, a média e o desvio padrão das aptidões e o número de avaliações, e identificados a melhor e pior aptidão da última iteração do algoritmo.
- Considerando que as cinco funções possuem mínimo global da função de avaliação *f(x) = 0*, foram estipuladas três condições de parada do

algoritmo: *i*) alcançar o mínimo global da função com precisão de 1e-10; ou *ii*) atingir o número máximo de 30.000 avaliações; ou *iii*) completar 10 iterações seguidas sem melhora na função objetivo.

- Quatro testes foram feitos para avaliar a adição de cada um dos principais módulos do PSO+. Além do PSO original, são avaliados o PSO+ com múltiplos enxames (ME), PSO+ com múltiplos enxames e operador de cruzamento aritmético (ME+RA) e PSO+ com múltiplos enxames, operador de cruzamento aritmético e pontos de apoio (ME+RA+PA).

- *Função esfera ou primeira função de De Jong*

Trata-se de uma das mais funções mais simples utilizadas em testes de algoritmos evolucionários. A mesma é contínua, convexa, unimodal, separável e pode ser ampliada para qualquer número de variáveis. É uma função do tipo quadrática, possuindo um único ótimo local (e, portanto, global) no ponto $o = (0, \dots, 0)$. Neste estudo de caso, o espaço de busca é definido em $[-600, 600]$ para cada variável de decisão. Sua forma analítica é dada por:

$$f_{sphere} = \sum_{i=1}^{n} x_i^2, \qquad (27)$$

onde *x* é vetor de decisão de n variáveis ou dimensões. Sua representação para o caso de duas variáveis é dada na Figura 15 para a esfera bidimensional ($f(x, y) = x^2 + y^2$).

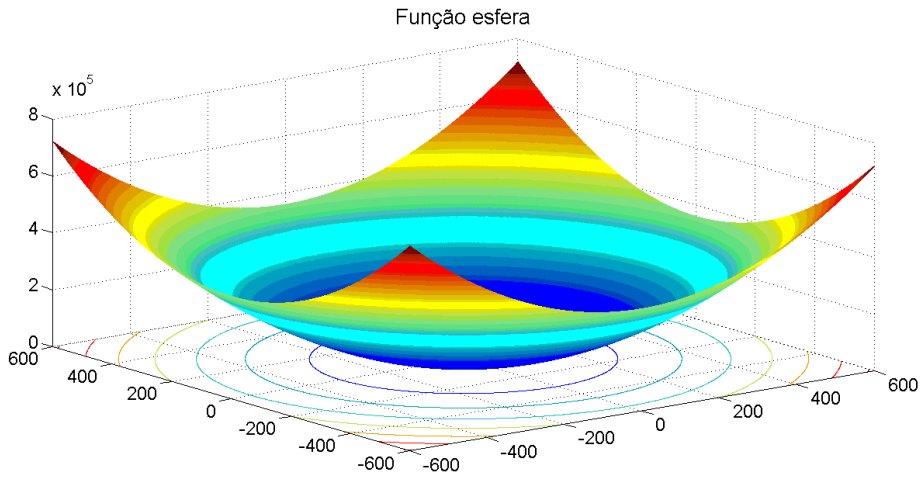


Figura 15. Mapa 3D da função esfera bidimensional.

- *Função de Ackley*

Trata-se de uma função multimodal, separável e com vários ótimos locais que no domínio [-30, 30] mais parecem com ruído, embora estejam localizados em intervalos regulares. A função de Ackley possui ótimo global no ponto $o = (0, \dots, 0)$, constitui um problema de complexidade moderada e sua forma analítica é dada por:

$$f_{Ackley} = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + \quad (28)$$
$$+20 + \exp(1),$$

onde $\boldsymbol{x}$ é vetor de decisão de n variáveis ou dimensões. Sua representação para o caso de duas variáveis é dada na Figura 16.

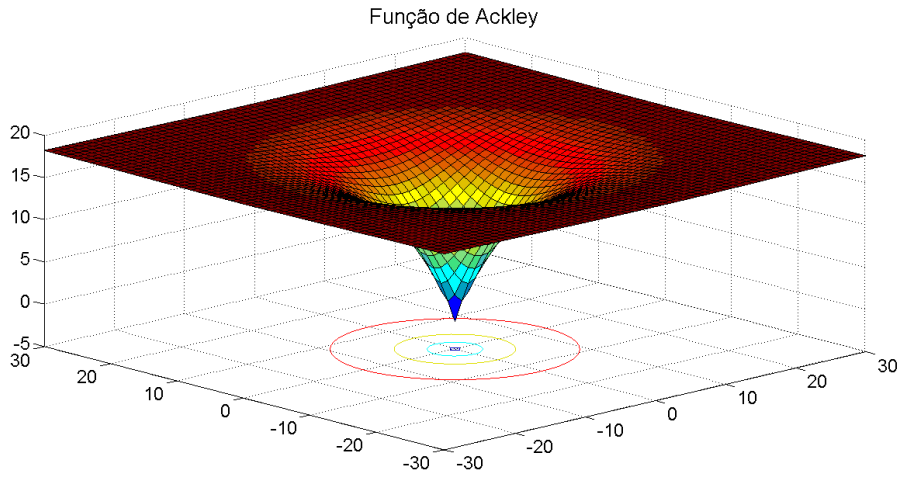


Figura 16. Mapa 3D da função de Ackley bidimensional

- *Função de Rastrigin*

É uma função não convexa, multimodal e separável. Possui vários ótimos locais arranjados em um grafo regular (*regular lattice*), com o ótimo global localizado no ponto $o = (0, \dots, 0)$. O espaço de busca é definido no intervalo [-5.12; 5.12] em cada variável. Constitui um problema razoavelmente difícil devido à grande quantidade de ótimos locais. Sua forma analítica é dada por:

$$f_{Rastrigin} = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10), \quad (29)$$

onde $\boldsymbol{x}$ é vetor de decisão de n variáveis ou dimensões. Sua representação para o caso de duas variáveis é dada na Figura 17.

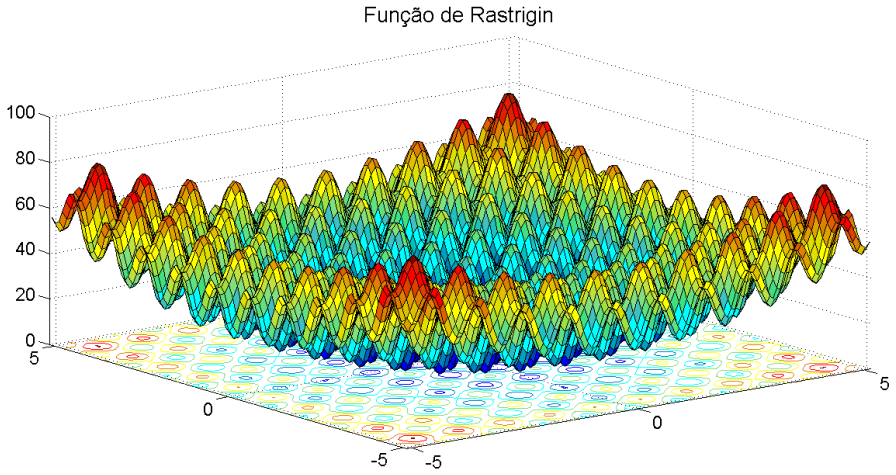


Figura 17. Mapa 3D da função de Rastrigin bidimensional

- *Função de Rosenbrock ou segunda função de De Jong*

Também conhecida como vale ou função banana de Rosenbrock, consiste em uma função não convexa, unimodal e não separável. O mínimo global está dentro de um vale plano, longo, estreito e em formato parabólico. Chegar ao vale é uma tarefa fácil; a dificuldade está em convergir para o mínimo global, localizado no ponto $o = (1, \dots, 1)$. Esta função também é quadrática e tem o seu domínio definido, neste estudo de caso, no intervalo de [-9, 11]. Sua forma analítica é dada por:

$$f_{Rosenbrock} = \sum_{i=1}^{n-1} (100(x_i^2 - x_{1+i})^2 + (x_i - 1)^2) \qquad (30)$$

onde $\boldsymbol{x}$ é vetor de decisão de n variáveis ou dimensões. Sua representação para o caso de duas variáveis é dada na Figura 18.

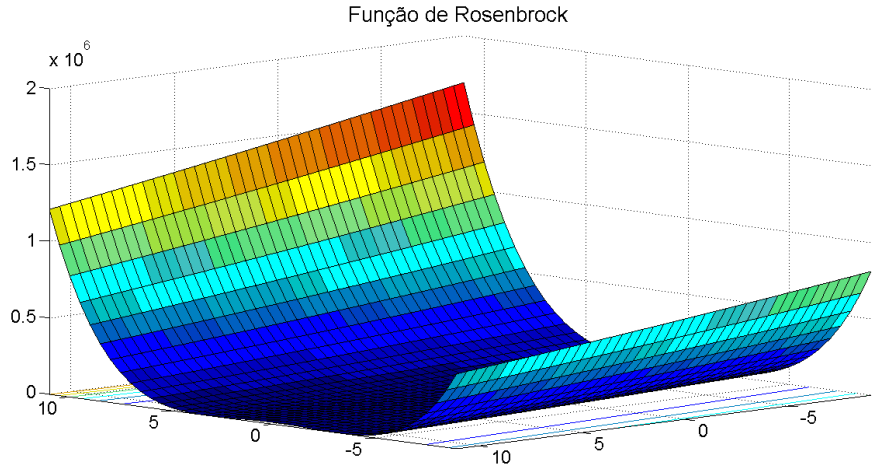


Figura 18. Mapa 3D da função Rosenbrock bidimensional

- *Função Griewank*

Esta função é multimodal e não separável, com vários ótimos locais dentro do espaço de busca, definido no intervalo [-30, 30]. É semelhante à função de Rastrigin, mas possui maior número de ótimos locais. Possui ótimo global localizado em $o = (1, \ldots, 1)$. A interpretação desta função modifica-se com a escala; a visão geral sugere função convexa, a visão de média escala sugere a existência de mínimos locais e, por fim, o exame mais próximo permite identificar uma estrutura complexa de inúmeros mínimos locais. Verifica-se na estrutura analítica, relacionada a seguir, que um termo de produto introduz interdependência entre as variáveis. O objetivo disto é fazer com que as técnicas que otimizam cada variável de forma independente falhem.

$$f_{Griewank} = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \qquad (31)$$

onde ***x*** é vetor de decisão de *n* variáveis ou dimensões. Sua representação para o caso de duas variáveis é dada na Figura 19.

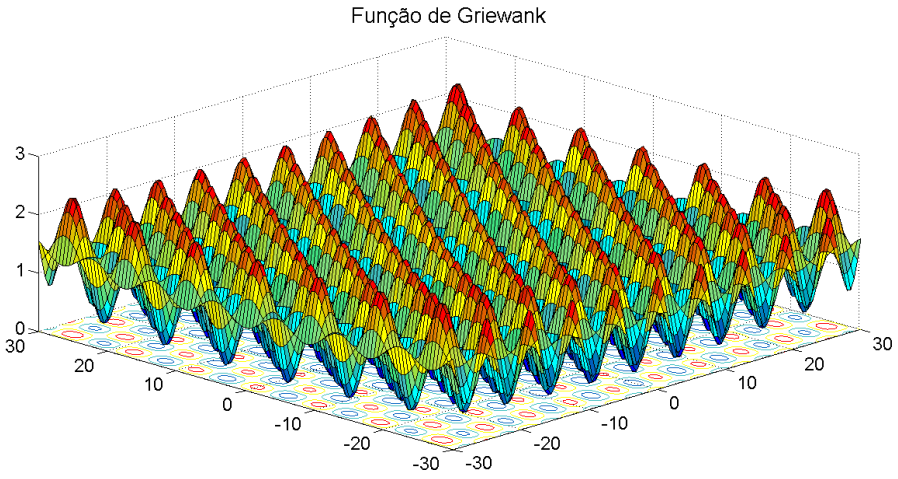


Figura 19. Mapa 3D da função de Griewank bidimensional

4.1.2. Resultados

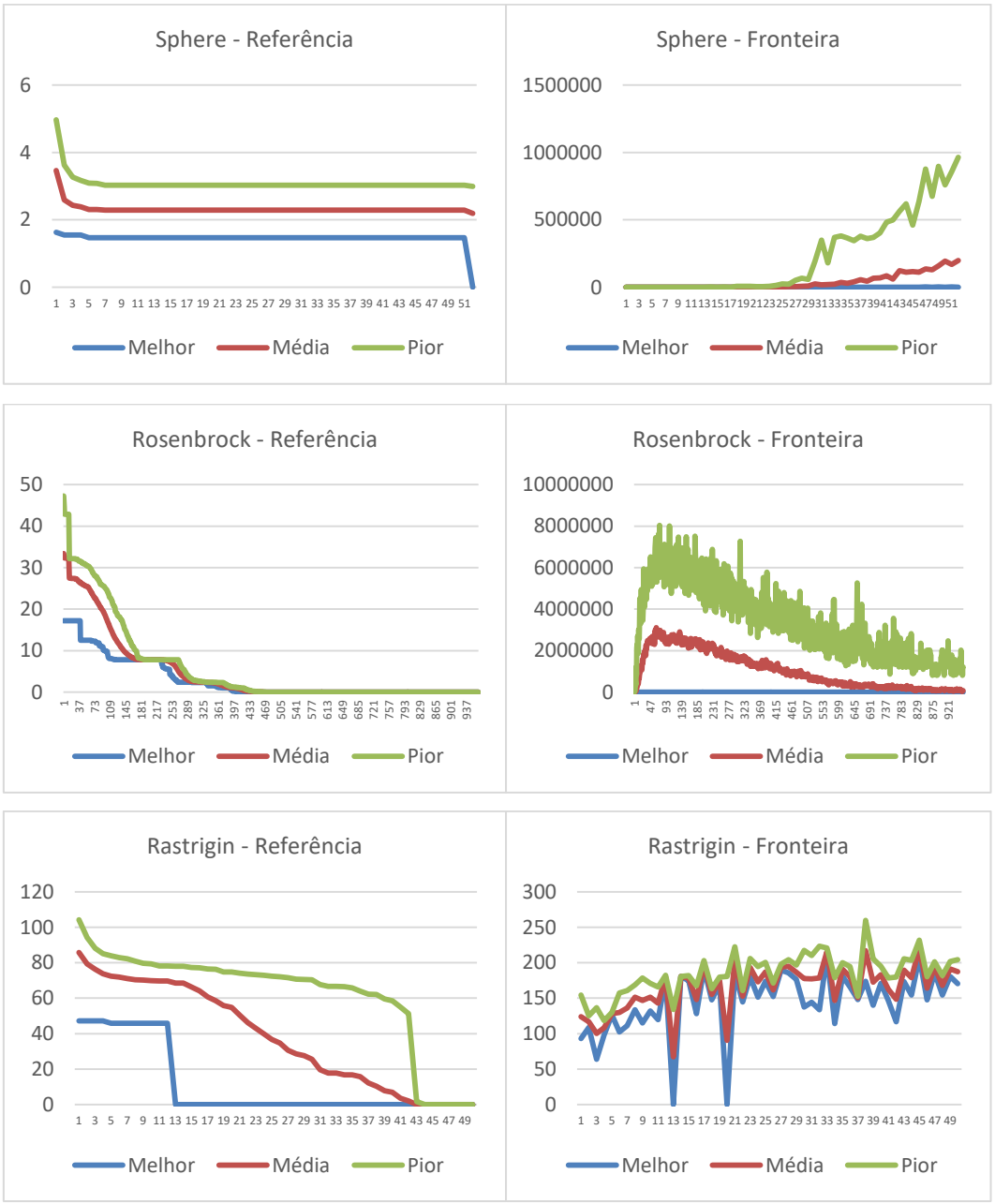
Nesta seção são comparados os módulos que sozinhos não fornecem ao algoritmo a capacidade de lidar com restrições lineares e não lineares, mas somente com restrições de domínio, assim como os módulos que fornecem essa capacidade ao algoritmo. Em função disso, nesse primeiro experimento, somente os *benchmarks* que possuem restrições de domínio são avaliados.

A Tabela 1 mostra os melhores resultados para os *benchmarks*, adicionando a cada teste uma nova funcionalidade do PSO+ (todas elas têm a mesma topologia de vizinhança definida no capítulo de modelagem do algoritmo: (i) PSO original; (ii) ME: PSO+ com múltiplos enxames; (iii) ME+RA: PSO+ com múltiplos enxames e redirecionamento aritmético; (iv) ME+RA+PA: PSO+ com múltiplos enxames, redirecionamento aritmético e pontos de apoio). O PSO+ completo, em geral, supera os testes anteriores em solução e em número de avaliações, exceto para a função Rosenbrock, onde o número de simulações até a convergência é muito maior que o das outras implementações (*). Como mencionado na seção anterior, o mínimo global desta função está dentro de um vale plano, longo, estreito e em formato parabólico, o que torna fácil a chegada ao vale, mas muito difícil a convergência para o mínimo global. Pode-se observar no gráfico da função que o enxame de fronteira, que mantém a diversidade do sistema, tende a aproximar a avaliação do melhor à do pior conforme o enxame de referência se aproxima do ótimo global.

Tabela 1. Melhores resultados para cada um dos módulos implementados no algoritmo de enxame de partículas aplicados à benchmarks com restrições de domínio somente.

Benchmark	Target	Algoritmo	Média dos Melhores	Avaliações
Sphere	0	Original	9.09E-7	3000
		ME	6.83E-6	2970
		ME+RA	1.27E-7	3000
		ME+RA+PA	**0**	**1560**
Rosenbrock	0	Original	2.38E-3	2280
		ME	2.91E-4	2580
		ME+RA	1.16E-3	3000
		ME+RA+PA	**0**	**28950***
Rastrigin	0	Original	31.81	2760
		ME	15.13	2910
		ME+RA	16.44	2700
		ME+RA+PA	**0**	**390**
Griewank	0	Original	4.28E-4	2220
		ME	3.80E-4	2700
		ME+RA	2.28E-4	2850
		ME+RA+PA	**0**	**510**
Ackley	0	Original	8.46E-4	7500
		ME	8.71E-4	5400
		ME+RA	2.12E-2	1800
		ME+RA+PA	**0**	**420**

Os gráficos da Figura 20 mostram a evolução do PSO+ para cada uma das cinco funções. Pode-se observar a evolução no tempo dos enxames de referência e de fronteira. Este pode conter partículas inválidas, que não são avaliadas, e internamente recebem um valor máximo da representação de um *double* (1.7976931348623157E+308). Este enxame contém toda a diversidade das populações e, na maioria dos problemas, a tendência é que esta diversidade aumente com o tempo. O enxame de referência contém a melhor partícula até o momento e apoia a busca por novas soluções válidas.



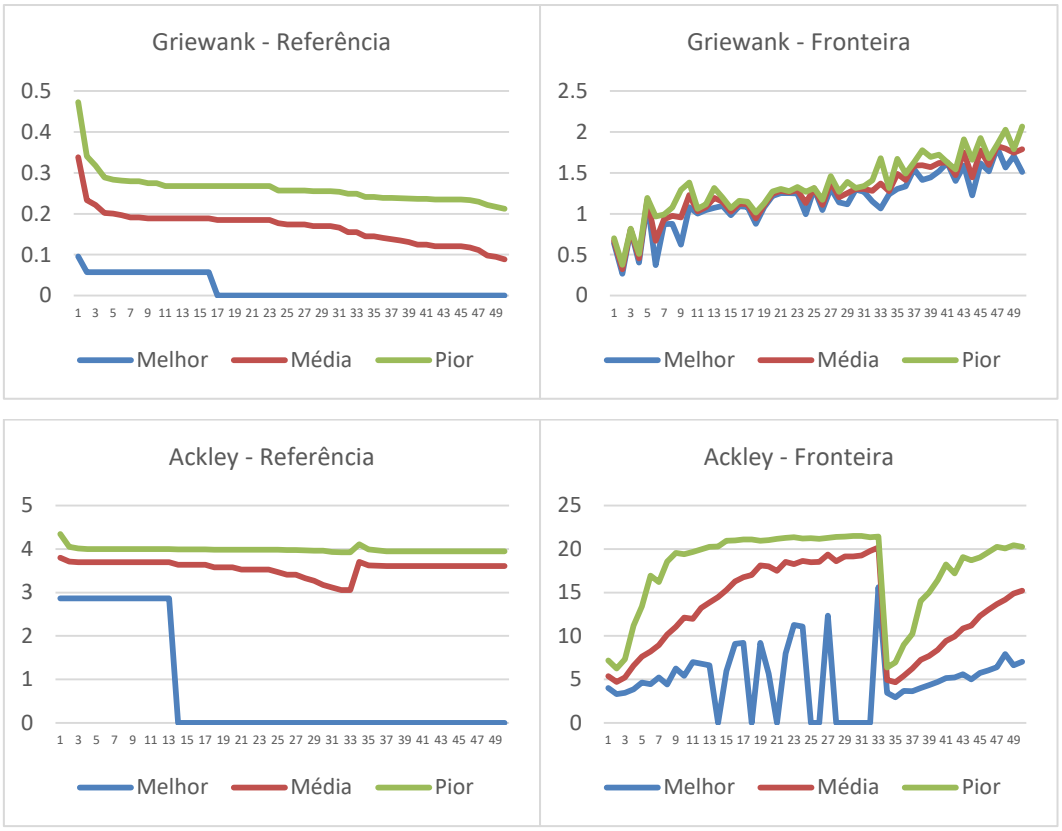


Figura 20. Gráficos de evolução das populações de referência e fronteira para as cinco funções benchmark.

4.2. Estudo de Caso com Funções *Benchmark* para Otimização com Restrições Lineares e Não Lineares

Neste estudo de caso, espera-se avaliar o desempenho do algoritmo proposto em problemas que envolvam restrições lineares e não lineares de igualdade e desigualdade em quantidades diversas.

As vinte e quatro funções *benchmark* para avaliação do modelo PSO+ proposto, todas elas criadas artificialmente (Liang *et al.*, 2006), estão descritas em detalhes no Apêndice A, e as suas características principais estão resumidas na Tabela 2, onde *D* é a dimensão do problema, *IL*, *IN*, *EL* e *EN* são os números de restrições para inequações lineares, inequações não lineares, equações lineares e equações não lineares, respectivamente, e *A* é o número de restrições ativas no ótimo global / número total de restrições do problema:

Tabela 2. Principais características das funções *benchmark.*

	Melhor Conhecido	D	Otimização	f(x)	$\|F\|/\|S\|$ (%)	IL	IN	EL	EN	A
G1	-15.0000	13	Min	quadrática	0.0111	9	0	0	0	6/9
G2	0.803619	20	Max	não linear	99.9971	1	1	0	0	1/2
G3	1	10	Max	polinomial	0.0020	0	0	0	1	1/1
G4	-30665.539	5	Min	quadrática	52.1230	0	6	0	0	2/6
G5	5126.4981	4	Min	cubic	0.0000	2	0	0	3	3/5
G6	-6961.81388	2	Min	cubic	0.0066	0	2	0	0	2/2
G7	24.306291	10	Min	quadrática	0.0003	3	5	0	0	6/8
G8	0.095825	2	Max	não linear	0.8560	0	2	0	0	0/2
G9	680.630057	7	Min	polinomial	0.5121	0	4	0	0	2/4
G10	7049.25	8	Min	linear	0.0010	3	3	0	0	6/6
G11	0.75	2	Min	quadrática	0.0000	0	0	0	1	1/1
G12	1	3	Max	quadrática	4.7713	0	9^3	0	0	$0/9^3$
G13	0.0539498	5	Min	exponencial	0.0000	0	0	0	3	3/3
G14	-47.7648884595	10	Min	não linear	0.0000	0	0	3	0	3/3
G15	961.7150222899	3	Min	quadrática	0.0000	0	0	1	1	2/2
G16	-1.9051552586	5	Min	não linear	0.0204	4	34	0	0	4/38
G17	8853.5396748064	6	Min	não linear	0.0000	0	0	0	4	4/4
G18	-0.8660254038	9	Min	quadrática	0.0000	0	13	0	0	0/13
G19	32.6555929502	15	Min	não linear	33.4761	0	5	0	0	0/5
G20	0.2049794002*	24	Min	linear	0.0000	0	6	2	12	16/20
G21	193.7245100700	7	Min	linear	0.0000	0	1	0	5	6/6
G22	236.4309755040	22	Min	linear	0.0000	0	1	8	11	19/20
G23	-400.0551000000	9	Min	linear	0.0000	0	2	3	1	6/6
G24	-5.5080132716	2	Min	linear	79.6556	0	2	0	0	2/2

*A melhor solução conhecida para o benchmark g20 é inválida para algumas restrições.

A razão $|F|/|S|$ foi determinada experimentalmente a partir da geração de um milhão de pontos aleatórios em S (espaço de busca que respeite restrições de domínio) e verificando-se se pertenciam ou não a F (espaço de busca válido). Como visto na Tabela 2, os vinte e quatro *benchmarks* representam um conjunto razoavelmente diverso de funções que auxiliarão na avaliação do modelo criado frente a outros.

4.2.1. Avaliação do PSO+

A Tabela 3 apresenta os melhores resultados para o PSO+ com e sem aplicação do conceito de pontos de apoio. A análise deste módulo em particular foi escolhida devido aos resultados da seção anterior, que demonstram que o mesmo tem um grande impacto no número de avaliações.

Pode-se observar que o uso dos pontos de apoio auxilia o algoritmo a convergir em problemas com restrições ativas no ótimo global. As condições de cada otimização feita seguem:

- Em cada função foram realizados 10 experimentos independentes, sendo calculados, ao final, a média e o desvio padrão das aptidões e o número de avaliações, e identificados a melhor e a pior aptidão da última iteração do algoritmo.
- Foram estipuladas três condições de parada do algoritmo: *i*) alcançar o mínimo global da função com precisão de 1e-10; ou *ii*) atingir o número máximo de 30.000 avaliações; ou *iii*) completar 10 gerações seguidas sem melhora na função objetivo.

A introdução dos pontos de apoio proporcionou a obtenção de melhores resultados para a grande maioria dos problemas. Todas as funções que obtiveram resultados melhores, com exceção de G18 e G19, envolvem restrições ativas nas soluções ótimas globais (o número de restrições ativas no ótimo global para cada função é mostrado na Tabela 2). Nas funções G1, G6, G8, G11, G12 e G24, os dois algoritmos chegaram ao mesmo resultado ótimo (G8 e G12 não possuem restrições ativas no ponto ótimo). Percebe-se também que, na implementação com pontos de apoio, a média dos melhores entre todos os 10 experimentos está mais estável, igual ou bem próxima ao ótimo, indicando uma robustez do algoritmo em encontrar o ótimo.

Além disso, a diversidade garantida pelo uso dos pontos de apoio permite que o PSO+ convirja mais rapidamente para um ótimo global em todas as funções, exceto na G11, em que a média de avaliações do algoritmo sem pontos de apoio foi consideravelmente menor (60 vs. 20790 avaliações).

O PSO+ conseguiu chegar ao ótimo global nos *benchmarks* G1 ao G16, G18 e G24. Obteve resultados melhores que o ótimo nos *benchmarks* G20, G21 e G23, o que pode ser explicado pelas tolerâncias utilizadas na conversão das restrições de igualdade em restrições de desigualdade descrita na seção 3.1. O algoritmo teve uma dificuldade maior de chegar à solução ótima nos *benchmarks* G17 e G19, funções essas que apresentam um grande número de restrições e um espaço de busca válido muito reduzido. Além disso, não foi possível encontrar soluções válidas para o *benchmark* G22. A provável causa está no tamanho do espaço de busca válido, que pode ser observado na Tabela 2 com a razão |F|/|S|=0.0000, determinada experimentalmente a partir da geração de um milhão de pontos aleatórios em S (espaço de busca que respeite restrições de domínio) e verificando-

se se pertencem ou não a F (espaço de busca válido). A Tabela 3 mostra as melhores soluções em negrito.

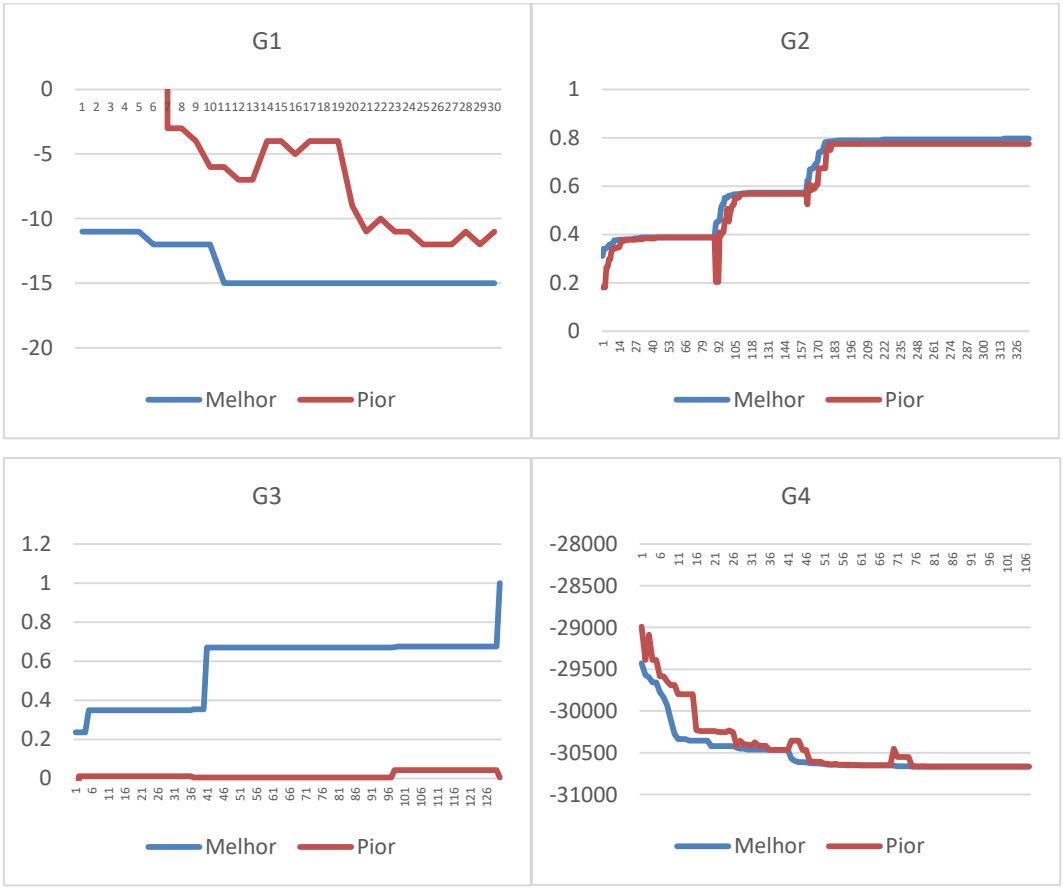
Tabela 3. Melhores resultados do PSO+

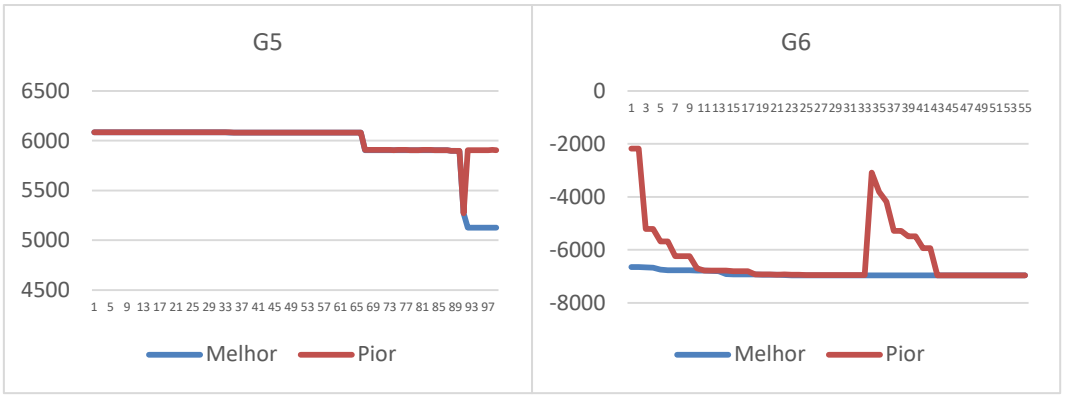
	Ótimo Conhecido	P.A	Melhor	Pior*	Média dos Melhores†	Desvio Padrão	Avaliações
G1	-15.0000	Não	**-15.000**	-15	-12	1.55	570
		Sim	**-15.000**	**-11**	**-15.0**	**0**	**330**
G2	0.803619	Não	0.6274	0.6272	0.5966	1.33e-15	30000
		Sim	**0.7967**	**0.7743**	**0.7967**	**1.59e-6**	**9510**
G3	1	Não	0.6176	0.4969	0.4776	5.55e-16	30000
		Sim	**1.0000**	**0.0939**	**0.6739**	**1.16e-15**	**3900**
G4	-30665.539	Não	-30649.4	-30648.4	-30563.17	2.91e-11	30000
		Sim	**-30665.5**	**-30665.5**	**-30661.09**	**5.09e-11**	**2190**
G5	5126.4981	Não	5936.5	5936.4	6075.77	9.83e-3	2760
		Sim	**5126.49**	**5905.7**	**5126.49**	**1.66e-10**	**2760**
G6	-6961.81388	Não	**-6961.8**	-6763.70	-6675.34	3.21e-12	5160
		Sim	**-6961.8**	**-6783.1**	**-6961.7**	**3.18e-1**	**2490**
G7	24.3062091	Não	29.44	48.89	56.78	1.5e-6	30000
		Sim	**24.310**	**42.25**	**26.46**	**8.52e-14**	**1860**
G8	0.095825	Não	**0.09582**	0.0910	0.0957	3.12e-2	3570
		Sim	**0.09582**	**0.08544**	**0.09582**	**1.89e-4**	**1200**
G9	680.6300573	Não	681.2795	684.49	684.28	1.15e-3	30000
		Sim	**680.6300**	**681.47**	**681.6134**	**9.45e-1**	**30000**
G10	7049.25	Não	7892.7588	9333.63	9099.17	201.91	30000
		Sim	**7050.2600**	**7222.77**	**7050.26**	**8.13e-3**	**28980**
G11	0.75	Não	**0.75**	**0.8027**	**0.75**	**8.88e-2**	**60**
		Sim	**0.75**	1.00	0.75	1.33e-15	20790
G12	1	Não	**1.0000**	0.74	0.9482	8.25e-3	150
		Sim	**1.0000**	**0.52**	**1.0000**	**1.27e-10**	**60**
G13	0.0539	Não	0.0609	0.3157	0.3018	3.82e-3	30000
		Sim	**0.0539**	**0.0622**	**0.0539**	**4.13e-17**	**2000**
G14	-47.7648884595	Não	-64.8281	-64.82	-64.82	4.66e-6	2970
		Sim	**-47.3639**	**-47.36**	**-47.36**	**6.38e-14**	**1650**
G15	961.7150222899	Não	966.4074	966.40	966.40	1.02e-12	4320
		Sim	**961.6500**	**961.65**	**961.65**	**1.72e-7**	**2010**
G16	-1.9051552586	Não	-1.8694	-1.8694	-1.8694	1.55e-15	23730
		Sim	**-1.9030**	**-1.9030**	**-1.9030**	**3.37e-8**	**3780**
G17	8853.5396748064	Não	8962.8282	8962.82	8962.82	5.45e-12	13710
		Sim	**8942.0700**	**8967.63**	**8942.07**	**3.59e-9**	**2220**
G18	-0.8660254038	Não	-0.8333	-0.8333	-0.8333	1.11e-15	21060
		Sim	**-0.8642**	**-0.8587**	**-0.8642**	**7.77e-16**	**6030**
G19	32.6555929502	Não	169.6920	169.69	169.69	1.70e-13	30000
		Sim	**41.1469**	**86.39**	**49.48**	**1.41e-14**	**17160**
G20	0.2049794002	Não	0.18*	0.18	0.18	0	10020
		Sim	**0.1093***	**0.1093**	**0.1093**	**2.77e-17**	**3810**
G21	193.7245100700	Não	315.2171	315.21	315.21	3.41e-13	30000
		Sim	**189.3736**	**512.74**	**189.37**	**4.54e-13**	**17160**
G22	236.4309755040	Não	-	-	-	-	-
		Sim	-	-	-	-	-
G23	-400.0551000000	Não	-400.8	-1.6018	81.6071	2.18e-15	12990
		Sim	**-401.09**	**-2.38**	**-5.0265**	**4.44e-15**	**4650**
G24	-5.5080132716	Não	**-5.5080**	-5.5080	-5.5080	7.99e-15	1200
		Sim	**-5.5080**	**-3.00**	**-5.5080**	**7.99e-15**	**960**

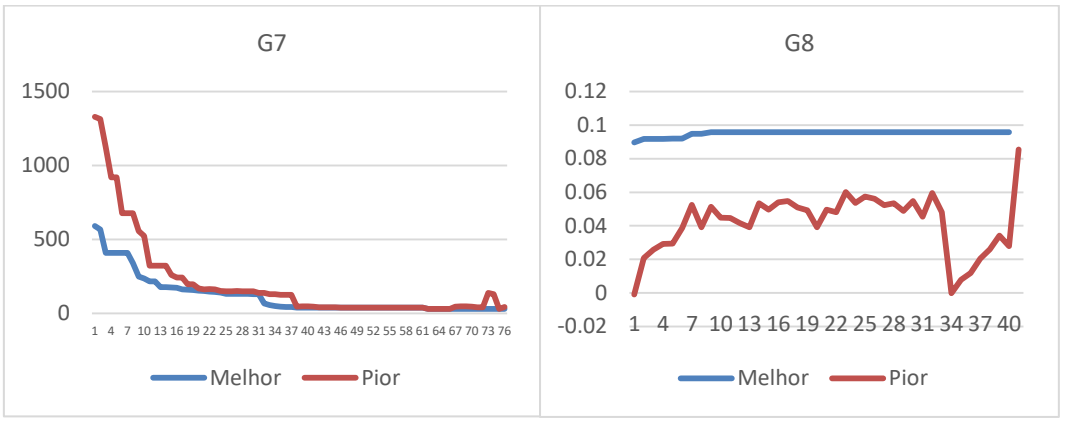
*A pior partícula está entre as partículas válidas do experimento com melhor desempenho. Soluções inválidas não são avaliadas. †A média é calculada a partir da média aritmética entre as melhores soluções dos 10 experimentos.

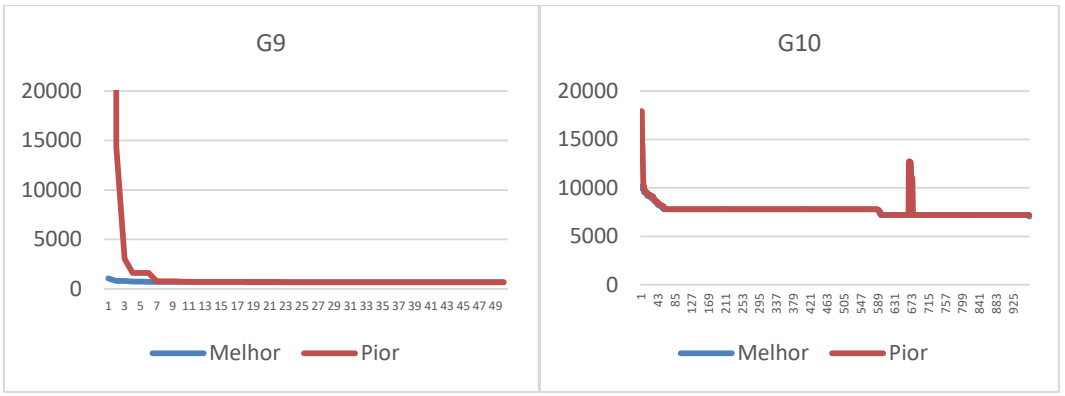
Os gráficos da evolução dos *benchmarks* são apresentados na Figura 21. Como o PSO+ é um algoritmo de otimização que opera com duas populações com tarefas e características específicas, os gráficos mostram a melhor partícula da população de referência – que mantém as melhores partículas válidas – a cada geração e a pior partícula da população de fronteira – que mantém partículas mais diversas e não necessariamente válidas, já que esse enxame só precisa garantir as restrições de domínio do problema – a cada geração. Partículas inválidas nunca serão avaliadas, não aparecendo, portanto, nos gráficos de evolução apresentados.

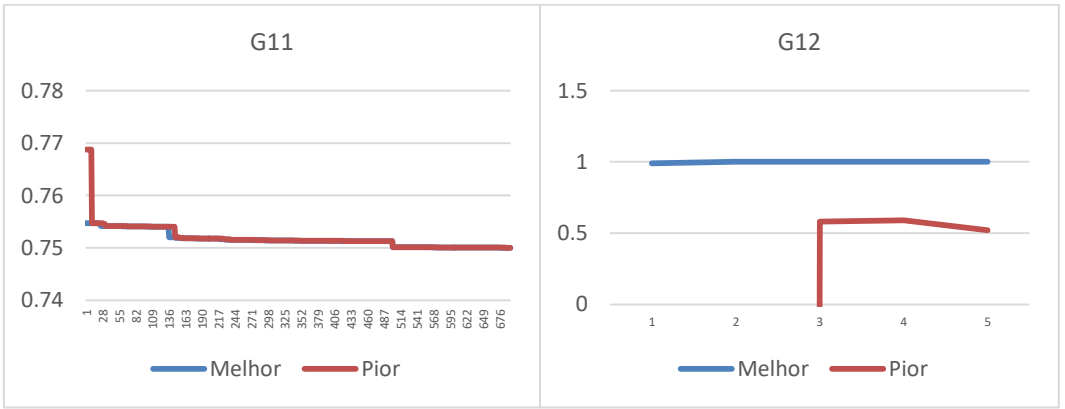
Apesar do uso de pontos de apoio para aumentar a diversidade da população e de uma topologia de vizinhança que, juntamente com os pontos de apoio, evita o problema de convergência prematura, pode-se observar que, aparentemente, alguns *benchmarks* não têm muita diversidade se comparadas a melhor com a pior partícula a cada geração. Isso pode ser devido a dois detalhes de implementação e do problema em si: (*i*) a população de fronteira mantém partículas inválidas e o algoritmo implementado não as avalia; (*ii*) alguns *benchmarks* têm um espaço de busca válido muito reduzido.

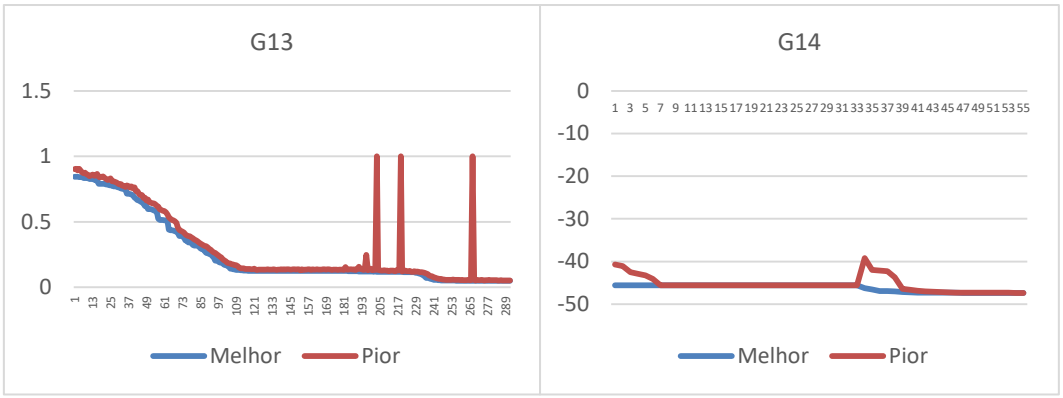


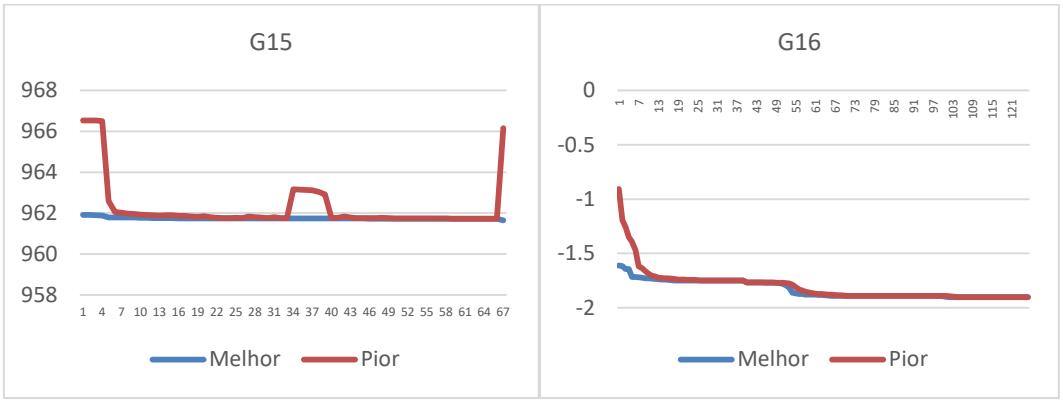


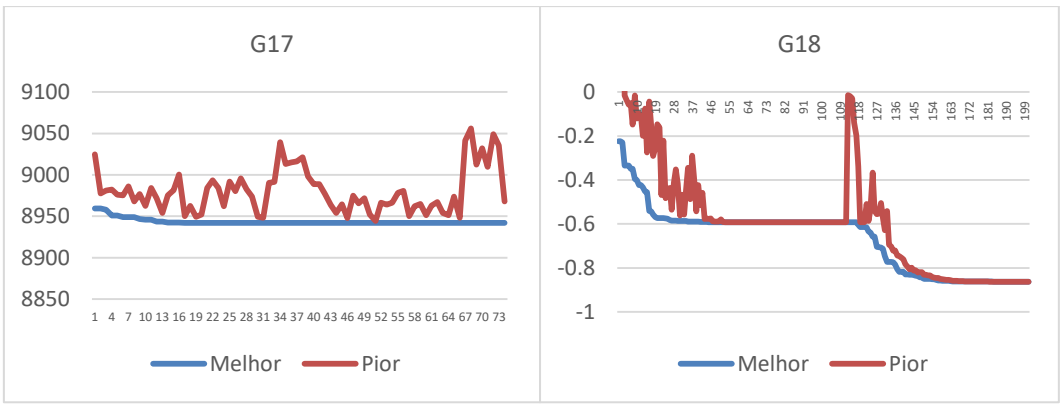


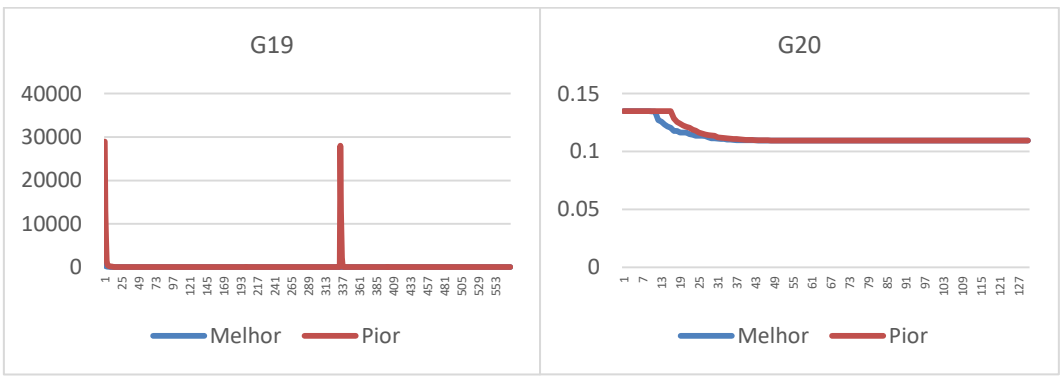












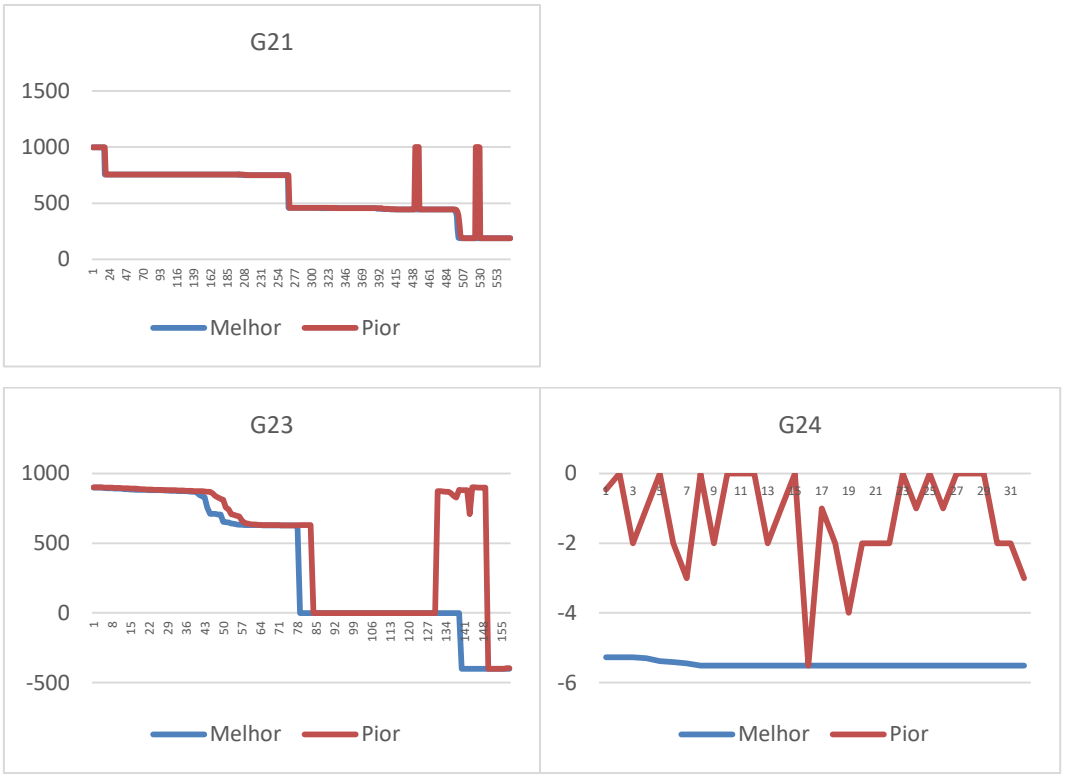


Figura 21. Gráficos de evolução da melhor é pior partículas dentre as duas populações (levando em conta somente soluções totalmente válidas).

4.2.2. Comparação dos Resultados com Outros Algoritmos PSO

A Tabela 4 compara os resultados do PSO+ com os de quatro outras implementações de PSO: PSO_Eb (Hu e Eberhart, 2002), Micro-PSO (Fuentes Cabrera e Coello Coello, 2007), PSO_CO (Li, Tian e Kong, 2005) e COPSO (Aguirre *et al.*, 2009).

No PSO_Eb, uma estratégia de preservação de viabilidade é empregada para lidar com restrições. Pode-se perceber que o PSO+ apresenta melhor desempenho – tanto em soluções ótimas quanto em número de simulações até a convergência – para todos os problemas apresentados ao PSO_Eb. Não há resultados para PSO_Eb aplicados aos problemas G02, G03, G5 e G13 ao G24.

O Micro-PSO é uma implementação para a solução de problemas com restrições que adotam um pequeno tamanho de população. O PSO+ supera o Micro-PSO para os problemas G1 e G4 ao G13, enquanto o Micro-PSO produz resultados superiores para G2 e G3. O trabalho não forneceu resultados de otimização para os problemas G14 ao G24.

O PSO_CO baseia-se no pressuposto de que qualquer solução válida é melhor do que qualquer solução inválida. Nos problemas G1, G4, G5, G6, G8, G9 e G11, ambos os algoritmos mostram comportamento semelhante. Não há resultados para PSO_CO aplicado às funções G02, G03, G7, G10, G12 ao G24. Só é possível avaliar o algoritmo em relação à solução ótima, já que o trabalho não fornece dados de número de avaliações até a convergência.

Finalmente, o COPSO utiliza operadores de perturbação para evitar convergência prematura e uma nova estrutura de vizinhança em anel. Além disso, a técnica de controle de restrições é baseada na validade das partículas e na soma das violações das restrições. O PSO+ supera esse algoritmo para os problemas G1, G3 ao G6, G8 ao G10, G12 ao G16, G21, G23 e G24 seja em solução ótima ou em número de avaliações até a convergência ao ótimo. Ambos algoritmos tiveram desempenho bem próximos em G2, G7, G11 e G18, mas o COPSO conseguiu superar o PSO+ em G17 e G19. Quanto ao número de avaliações do algoritmo em relação ao PSO+, percebe-se que ele é de uma ordem de grandeza maior: da ordem de 10^5 para o COPSO e, para o PSO+, 10^3 avaliações para G17 e 10^4 para G19, o que torna a comparação não muito justa. O PSO+ não foi capaz de encontrar soluções válidas em G22.

Tabela 4. Comparação de resultados entre diferentes implementações do PSO

	Ótimo Conhecido		**Melhor**	**Média dos Melhores**	**Pior**	**Desvio Padrão**	**Avaliações**
G1	-15.0000	PSO+	**-15.000**	**-15.0**	**-11**	**0**	**330**
		PSO_Eb	-15.000	-	-	-	25000
		Micro-PSO	-15.000	-13.2734	-	-	240000
		PSO_CO	-15.000	-15.00	-	-	-
		COPSO	-15.000	-15.000	-15.000	0	90800
G2	0.803619	PSO+	0.7967	0.7967	0.7743	1.59e-6	**9510**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	0.8036	0.78	-	-	240000
		PSO_CO	-	-	-	-	-
		COPSO	**-0.8036**	**-0.8013**	**-0.7865**	**4.5e-3**	**142900**
G3	1	PSO+	**1.0000**	**0.6739**	**0.0939**	**1.16e-15**	**3900**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	**1.0004**	0.9936	-	-	240000
		PSO_CO	-	-	-	-	-
		COPSO	-1.0000	-1.0000	-1.0000	3.15e-7	315100
G4	-30665.539	PSO+	**-30665.5**	**-30661.09**	**-30665.5**	**5.09e-11**	**2190**
		PSO_Eb	-30665.5	-	-	-	25000
		Micro-PSO	-30665.5	-30665.53	-	-	240000
		PSO_CO	-30665.5	-30665.59	-	-	-
		COPSO	-30665.5	-30665.5	-30665.5	0	59600

G5	5126.4981	PSO+	**5126.49**	**5126.49**	**5905.7**	**1.66e-10**	**2760**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	5126.6467	5495.23	-	-	240000
		PSO_CO	5126.49	5129.30	-	-	-
		COPSO	5126.49	5126.49	5126.49	0	315100
G6	-6961.81388	PSO+	**-6961.8**	**-6961.7**	**-6783.1**	**3.18e-1**	**2490**
		PSO_Eb	-6961.7	-	-	-	25000
		Micro-PSO	-6961.7	-6961.83	-	-	240000
		PSO_CO	-6961.8	-6961.81	-	-	-
		COPSO	-6961.8	-6961.8	-6961.8	0	47100
G7	24.306209	PSO+	24.310	26.46	42.25	8.52e-14	**1860**
		PSO_Eb	24.4420	-	-	-	25000
		Micro-PSO	24.3278	24.6996	-	-	240000
		PSO_CO	-	-	-	-	-
		COPSO	**24.306**	**24.306**	**24.306**	**3.3e-6**	**185500**
G8	0.095825	PSO+	**0.0958**	**0.09582**	**0.08544**	**1.89e-4**	**1200**
		PSO_Eb	0.0958	-	-	-	25000
		Micro-PSO	0.0958	0.0958	-	-	240000
		PSO_CO	0.0958	0.0959	-	-	-
		COPSO	0.0958	0.0958	0.0958	0	3600
G9	680.630057	PSO+	**680.6300**	**681.6134**	**681.47**	**9.45e-1**	**30000**
		PSO_Eb	680.6570	-	-	-	25000
		Micro-PSO	680.6307	680.64	-	-	240000
		PSO_CO	680.6300	680.65	-	-	-
		COPSO	680.6300	680.6300	680.6300	0	69900
G10	7049.25	PSO+	7050.2600	**7050.26**	**7222.77**	**8.13e-3**	28980
		PSO_Eb	7131.0100	-	-	-	25000
		Micro-PSO	7090.4524	7747.63	-	-	240000
		PSO_CO	-	-	-	-	-
		COPSO	**7049.2486**	**7049.2500**	**7049.26**	**3.6e-3**	**167200**
G11	0.75	PSO+	**0.7500**	**0.75**	**1.00**	**1.33e-15**	**20790**
		PSO_Eb	0.7500	-	-	-	25000
		Micro-PSO	**0.7499**	**0.7673**	-	-	**240000**
		PSO_CO	0.7499	0.7499	-	-	-
		COPSO	0.7499	0.7499	0.7499	**0**	315000
G12	1	PSO+	**1.0000**	**1.0000**	**0.52**	**1.27e-10**	**60**
		PSO_Eb	1.0000	-	-	-	25000
		Micro-PSO	1.0000	1.00	-	-	240000
		PSO_CO	-	-	-	-	-
		COPSO	1.0000	1.0000	1.0000	0	400
G13	0.0539	PSO+	**0.0539**	**0.0539**	**0.0622**	**4.13e-17**	**2000**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	0.0594	0.81	-	-	240000
		PSO_CO	-	-	-	-	-
		COPSO	0.0539	0.0539	0.0539	2.7e-6	315100
G14	-47.764888	PSO+	-47.3639	-47.36	-47.36	6.38e-14	1650
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	**-47.7611**	**47.7414**	**-47.6709**	**2.1e-6**	**326900**
G15	961.715022	PSO+	**961.6500**	**961.65**	**961.65**	**1.72e-7**	**2010**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-

		COPSO	**961.7151**	**961.7151**	**961.7151**	**0**	**315100**
G16	-1.905155	PSO+	-1.9030	-1.9030	-1.9030	3.37e-8	3780
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	**-1.9051**	**-1.9051**	**-1.9051**	**0**	**37200**
G17	8853.539674	PSO+	8942.0700	8942.07	8967.63	3.59e-9	**2220**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	**8856.5023**	**8877.8128**	**8941.34**	**30.11**	**315100**
G18	-0.866025	PSO+	-0.8642	-0.8642	-0.8587	7.77e-16	**6030**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	**-0.8660**	**-0.8660**	**-0.8655**	**8.7e-5**	**102200**
G19	32.655592	PSO+	41.1469	**49.48**	**86.39**	**1.41e-14**	**17160**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	**32.3496**	**32.4115**	**32.5715**	**6.3e-2**	**206800**
G20	0.204979	PSO+	**0.1093***	**0.1093**	**0.1093**	**2.77e-17**	**3810**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	0.2040*	0.2120	0.2332	6.9e-3	-
G21	193.724510	PSO+	**189.3736**	**189.37**	**512.74**	**4.54e-13**	**17160**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	205.8526	273.2980	303.4548	23.85	-
G22	236.430975	PSO+	-	-	-	-	-
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	**157.51***	**5376.22**	**18732.78**	**5.0e-3**	**NR**
G23	-400.055100	PSO+	**-401.09**	**-5.0265**	**-2.38**	**4.44e-15**	**4650**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	-361.8566	-136.5642	3.7757	84.52	-
G24	-5.508013	PSO+	**-5.5080**	**-5.5080**	**-3.00**	**7.99e-15**	**960**
		PSO_Eb	-	-	-	-	-
		Micro-PSO	-	-	-	-	-
		PSO_CO	-	-	-	-	-
		COPSO	-5.5080	-5.5080	-5.5080	0	14900

4.2.3. Comparação dos Resultados com Outros Algoritmos Evolucionários

A Tabela 5 mostra os resultados obtidos com o PSO+ e com outros algoritmos evolucionários. As três primeiras implementações, RGenIII, GenV e RGenV (Kato,

Sakawa e Katagiri, 2008) são implementações revisadas do Genocop III com melhorias no tempo de processamento e em algumas limitações. O PSO+ supera todas estas implementações revisadas tanto em solução quanto em número de avaliações. Não há resultados para RGEnIII, GenV e RGEnV aplicados aos problemas G02 ao G05, G08, G12 ao G24.

Colônia Artificial de Abelhas (ABC, do inglês *Artificial Bee Colony*) é um algoritmo de inteligência de enxames relativamente novo que tem mostrado desempenho competitivo quando comparado a outros algoritmos baseados em população. O algoritmo eABC (Ahmad e Babaeizadeh, 2014) é proposto para resolver problemas de otimização com restrições. No processo de otimização, os indivíduos são inicializados aleatoriamente e a evolução é feita utilizando penalização competitiva. O PSO+ supera-o em resultados de otimização e número de avaliações para os *benchmarks* G1, G3, G4, G6 ao G10, G12 ao G15, G18 e G24. O desempenho do eABC é ligeiramente melhor para G2 e G16, e consideravelmente melhor em G17 e G19. Isto pode ser devido ao mesmo motivo que levou o algoritmo COPSO, da seção anterior, a superar o PSO+ nesses dois *benchmarks*: o número de avaliações é de uma ordem de grandeza maior nesses testes. Em G5, G11 e G23 o PSO+ chegou ao ótimo, mas o eABC superou o ótimo, e isso se deve à diferença na tolerância usada para as restrições lineares e não lineares de igualdade. Não foram reportados resultados para G20, G21 e G22.

A meta-heurística de Vaga-Lume (FA, do inglês *Firefly Algorithm*) é baseada no comportamento natural dos vaga-lumes e no fenômeno de bioluminescência. Para lidar com restrições, Deshpande (Deshpande, Phatnani e Kulkarni, 2013) propôs um método de penalização competitiva. O PSO+ superou FA em todos os benchmarks, exceto G11, onde o resultado do FA foi melhor do que o ótimo, provavelmente devido às tolerâncias estabelecidas para as restrições de igualdade. Não são reportados resultados para G2, G3, G5, G10, G13 ao G24.

SSO-C (do inglês *Social Spider Optimization with Constraints*) (Cuevas e Cienfuegos, 2014) é baseado na simulação do comportamento cooperativo de aranhas. Nesse algoritmo, os indivíduos emulam um grupo de aranhas que interagem entre si de acordo com as leis de cooperação de colônias cooperativas. O algoritmo considera dois tipos diferentes de agentes: machos e fêmeas. Dependendo do gênero, cada indivíduo é conduzido por uma série de operadores. A inicialização dos indivíduos é feita de forma aleatória e para lidar com restrições é utilizado o

método de penalização competitiva. O PSO+ apresentou resultados melhores tanto em solução da otimização quanto em número de simulações para todos os *benchmarks* avaliados, exceto G2, onde o SSO-C superou o PSO+, e G11, em que o algoritmo obteve um resultado melhor que o ótimo pelo mesmo motivo do ocorrido com o algoritmo FA, onde a tolerância definida para as equações de igualdade foi menor do que a usada neste trabalho. Não foram reportados resultados para G1, G5, G8 ao G10, G12, G14, G16 ao G24.

Tabela 5. Comparação dos resultados com outros algoritmos evolucionários

	Ótimo Conhecido		**Melhor**	**Média dos Melhores**	**Pior**	**Desvio Padrão**	**Avaliações**
G01	15.0000	PSO+	**-15.000**	**-15.0**	**-11**	**0**	**330**
		RGen.III	-14.890	-14.89	-14.89		350000
		Gen.V	-15.000	-9.68	-2.15		350000
		RGen.V	-15.000	-12.88	-9.50		350000
		eABC	-15.000	-15.000	-15.000	0	240000
		FA	-	-	-	-	-
		SSO-C	-15.000	-15.000	-15.000	0	25000
G02	0.803619	PSO+	0.7967	0.7967	0.7743	1.59e-6	9510
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	0.8036	0.8021	0.7990	1.26e-3	240000
		FA	-	-	-	-	-
		SSO-C	**0.8036**	0.8015	0.7925	**3.5e-5**	**25000**
G03	1	PSO+	**1.0000**	**0.6739**	**0.0939**	**1.16e-15**	**3900**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	**-1.0041**	-1.0031	-1.0010	1.33e-3	240000
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G04	-30665.539	PSO+	**-30665.5**	**-30661.09**	**-30665.5**	**5.09e-11**	**2190**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	-30665.5	-30665.5	-30665.5	0	240000
		FA	-30665.0	-30664.67	-30663.9	-	12000
		SSO-C	-30665.5	-30665.53	-30665.14	1.1e-4	25000
G05	5126.4981	PSO+	**5126.49**	**5126.49**	**5905.7**	**1.66e-10**	**2760**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	**5126.39**	**5299.67**	**5968.58**	**248.42**	**240000**
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G06	-6961.81388	PSO+	**-6961.8**	**-6961.7**	**-6783.1**	**3.18e-1**	**2490**
		RGen.III	-6961.8	-6961.8	-6961.8		350000
		Gen.V	-6873.6	-6787.49	-6614.13		350000
		RGen.V	-6961.8	-6961.79	-6961.77		350000
		eABC	-6961.8	-6961.81	-6961.81	0	240000
		FA	-6956.6	-6956.63	-6953.47	-	12000
		SSO-C	-6961.8	-6961.01	-6961.91	1.1e-3	25000
G07	24.3062091	PSO+	24.310	26.46	42.25	8.52e-14	1860
		RGen.III	25.57	28.23	35.25		350000

		Gen.V	24.62	27.14	35.79		350000
		RGen.V	24.78	25.31	25.61		350000
		eABC	24.55	24.80	25.10	1.2e-1	240000
		FA	24.38	24.47	24.60	-	50000
		SSO-C	**24.306**	**24.306**	**24.306**	**4.9e-5**	**25000**
G08	0.095825	PSO+	**0.09582**	**0.09582**	**0.08544**	**1.89e-4**	**1200**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	0.09582	0.09582	0.09582	0	240000
		FA	**0.09582**	**0.09582**	**0.09582**	-	**12000**
		SSO-C	-	-	-	-	-
G09	680.630057	PSO+	**680.6300**	**681.6134**	**681.47**	**9.45e-1**	**30000**
		RGen.III	680.6300	680.63	680.63		350000
		Gen.V	680.6500	680.67	680.69		350000
		RGen.V	680.6300	680.85	680.98		350000
		eABC	680.6387	680.6512	680.6776	885.15	240000
		FA	680.8463	681.0415	681.2603	-	12000
		SSO-C	-	-	-	-	-
G10	7049.25	PSO+	**7050.2600**	**7050.26**	**7222.77**	**8.13e-3**	**28980**
		RGen.III	7129.0000	7508.32	7796.21		350000
		Gen.V	7255.54	8559.57	11597.24		350000
		RGen.V	7068.44	7182.63	7273.53		350000
		eABC	7304.817	7445.860	1647.175	87.75	**240000**
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G11	0.75	PSO+	**0.7500**	**0.75**	**1.00**	**1.33e-15**	**20790**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	0.7490*	0.7490	0.7490	5.0e-7	240000
		FA	**0.7490***	**0.7490**	**0.7490**	-	**12000**
		SSO-C	0.7499*	0.7499	0.7499	4.1e-9	25000
G12	1	PSO+	**1.0000**	**1.0000**	**0.52**	**1.27e-10**	**60**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	1.0000	1.0000	1.0000	0	240000
		FA	0.9999	0.9999	0.9999	-	12000
		SSO-C	-	-	-	-	-
G13	0.0539	PSO+	**0.0539**	**0.0539**	**0.0622**	**4.13e-17**	**2000**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	0.4910	0.9439	1.3252	1.71e-1	240000
		FA	-	-	-	-	-
		SSO-C	0.0539	0.0539	0.0539	6.3e-9	25000
G14	-47.764888	PSO+	**-47.3639**	**-47.36**	**-47.36**	**6.38e-14**	**1650**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	-45.97	-42.20	-39.11	1.46	240000
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G15	961.715022	PSO+	**961.6500**	**961.65**	**961.65**	**1.72e-7**	**2010**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	940.1215	957.7468	970.6761	7.75	240000
		FA		-	-	-	-
		SSO-C	961.9821	961.9987	962.0078	1.2	25000
G16	-1.905155	PSO+	**-1.9030**	**-1.9030**	**-1.9030**	**3.37e-8**	**3780**
		RGen.III	-	-	-	-	-

		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	**-1.9051**	**-1.9051**	**-1.9051**	**0**	**240000**
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G17	8853.539674	PSO+	8942.0700	8942.07	8967.63	3.59e-9	2220
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	**8871.616**	9052.462	9259.310	**1.32**	**240000**
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G18	-0.866025	PSO+	**-0.8642**	**-0.8642**	**-0.8587**	**7.77e-16**	**6030**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	-0.8424	-0.7418	0.0616	6.16e-2	240000
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G19	32.655592	PSO+	41.1469	49.48	86.39	1.41e-14	17160
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	**37.0839**	39.7023	42.4724	**1.36**	**240000**
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G20	0.204979	PSO+	**0.1093**	**0.1093**	**0.1093**	**2.77e-17**	**3810**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	-	-	-	-	-
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G21	193.724510	PSO+	**189.3736**	**189.37**	**512.74**	**4.54e-13**	**17160**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	-	-	-	-	-
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G22	236.430975	PSO+	-	-	-	-	-
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	-	-	-	-	-
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G23	-400.055100	PSO+	**-401.09**	**-5.0265**	**-2.38**	**4.44e-15**	**4650**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	**-704.385***	**-221.191**	**57.88**	**196.76**	-
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-
G24	-5.5080132	PSO+	**-5.5080**	**-5.5080**	**-3.00**	**7.99e-15**	**960**
		RGen.III	-	-	-	-	-
		Gen.V	-	-	-	-	-
		RGen.V	-	-	-	-	-
		eABC	-5.5080	-5.5080	-5.5080	0	240000
		FA	-	-	-	-	-
		SSO-C	-	-	-	-	-

4.2.4.
Número de Avaliações

A Tabela 6 apresenta o número de avaliações necessário para o PSO+ convergir e o número total de avaliações para os outros algoritmos (o número exato de avaliações até a convergência não está disponível; não há nenhuma informação para PSO_CO).

O PSO+ convergiu, para a maioria dos *benchmarks*, com muito menos avaliações. No caso do G10, PSO_Eb apresentou melhor desempenho em termos de número de avaliações, mas deve-se recordar (Tabela 4) que o PSO+ superou com folga o PSO_Eb em termos de resultados. Para G9 e G11, o PSO+ foi superado pelo algoritmo FA em termos de número de avaliações, sendo que o PSO+ superou o FA em termos de resultado para G9; já para G10, o FA apresentou um resultado melhor do que o ótimo em função das tolerâncias escolhidas para a solução do problema.

Tabela 6. Número de avaliações

	PSO+	PSO_Eb	Micro-PSO	PSO_CO	COPSO	RGen.III, Gen.V and RGen.V	eABC	FA	SSO-C
G01	**330**	25000	240000	-	90800	350000	240000	-	25000
G02	**9510**	-	240000	-	142900	-	240000	-	25000
G03	**3900**	-	240000	-	315100	-	240000	-	-
G04	**2190**	25000	240000	-	59600	-	240000	12000	25000
G05	**2760**	-	240000	-	315100	-	240000	-	-
G06	**2490**	25000	240000	-	47100	350000	240000	12000	25000
G07	**1860**	25000	240000	-	185500	350000	240000	50000	25000
G08	**1200**	25000	240000	-	3600	-	240000	12000	-
G09	30000	25000	240000	-	69900	350000	240000	**12000**	-
G10	28980	**25000**	240000	-	167200	350000	240000	-	-
G11	20790	25000	240000	-	315000	-	240000	**12000**	25000
G12	**60**	25000	240000	-	400	-	240000	12000	-
G13	**2000**	-	240000	-	315100	-	240000	-	25000
G14	**1650**	-	-	-	326900	-	240000	-	-
G15	**2010**	-	-	-	315100	-	240000	-	25000
G16	**3780**	-	-	-	37200	-	240000	-	-
G17	**2220**	-	-	-	315100	-	240000	-	-
G18	**6030**	-	-	-	102200	-	240000	-	-
G19	**17160**	-	-	-	206800	-	240000	-	-
G20	**3810**	-	-	-	NR	-	-	-	-
G21	**17160**	-	-	-	NR	-	-	-	-
G22	-	-	-	-	NR	-	-	-	-
G23	**4650**	-	-	-	NR	-	240000	-	-
G24	**960**	-	-	-	14900	-	240000	-	-

4.2.5.
Testes Estatísticos

O teste *Wilcoxon Signed Rank* (Derrac *et al.*, 2011) serve para responder a seguinte questão: duas amostras representam duas populações diferentes? Este é um método não paramétrico para comparação de duas amostras pareadas para detecção de diferenças significativas entre duas amostras, isto é, o comportamento de dois

algoritmos. O valor p-value calculado indica se a hipótese nula pode ser rejeitada e, consequentemente, que os diferentes resultados obtidos pelos dois modelos podem ser considerados estatisticamente significativos.

Na Tabela 7 são apresentados os resultados dos testes comparando os algoritmos dois a dois – para fins de comparação são utilizados os erros percentuais relativos de cada algoritmo para cada *benchmark* –, onde R^+ é a soma das classificações para os problemas nos quais o primeiro algoritmo (PSO+) superou o segundo, e R^- a soma das classificações para o oposto. Classificações iguais são ignoradas e são contabilizadas (porém, são mostradas na tabela). Note que a equação $R^+ + R^- + I = n * (n +1)/2$ deve ser verdadeira. Observando o *p-value* podemos ou não rejeitar a hipótese nula de que a média da diferença entre os modelos é zero, significando que um algoritmo superou o outro. Na tabela pode-se perceber uma melhora do PSO+ em relação ao PSO_Eb, Micro-PSO, RGen.III, Gen.V, RGen.V e FA. O *p-value* para os outros quatro algoritmos assumiram valores maiores que o limiar estabelecido (>0.2), o que significa que os diferentes resultados obtidos não podem ser considerados estatisticamente significativos.

Tabela 7. Resultados do teste de Wilcoxon.

Comparação	**R^+**	**R^-**	**I**	***p-value***
PSO+ versus PSO_Eb	10	0	290	0.06789
PSO+ versus Micro-PSO	37	8	255	0.08583
PSO+ versus PSO_CO	1	0	299	> 0.2
PSO+ versus COPSO	43	48	209	> 0.2
PSO+ versus RGen.III	6	0	294	0.1088
PSO+ versus Gen.V	10	0	290	0.06789
PSO+ versus RGen.V	3	0	297	0.1797
PSO+ versus eABC	100	36	164	> 0.2
PSO+ versus FA	25	3	272	0.06298
PSO+ versus SSO-C	4	6	290	> 0.2

4.3.
Estudo de Caso Real

4.3.1.
Otimização de Alocação de Poços

Um problema real de otimização não linear com restrições não lineares é a otimização de malhas de drenagens. Algoritmos de otimização podem ser aplicados, por exemplo, para maximizar o Valor Presente Líquido (VPL – descrito no Apêndice B) ou o óleo acumulado recuperado. Estes algoritmos podem ser utilizados para otimizar a localização, os tipos (injetor ou produtor), as trajetórias e a quantidade de novos poços. Esse tipo de otimização é geralmente custoso do ponto de vista computacional em virtude da quantidade de simulações necessárias.

Em ambiente *offshore*, os custos de perfuração e completação dos poços são extremamente elevados e a preocupação com a otimização do plano de desenvolvimento é vital para a viabilidade do projeto. A dificuldade com a alocação de poços é o elevado número de variáveis. Além disso, a interação entre essas variáveis é bastante complexa e não linear.

O objetivo então é, a partir da descrição de um reservatório dada pelo modelo de simulação, otimizar simultaneamente a localização, a trajetória, a quantidade e o tipo dos poços de forma a maximizar o valor presente líquido (VPL) do projeto, respeitando todas as restrições inerentes ao problema.

Diferentes algoritmos estocásticos têm sido aplicados ao problema de otimização de localização de poços: algoritmos genéticos (Emerick *et al.*, 2009), otimização por enxame de partículas (Cheng *et al.*, 2012; Onwunalu e Durlofsky, 2010) e perturbação simultânea com aproximação estocástica (Bangerth *et al.*, 2006). Abordagens determinísticas também têm sido utilizadas (Sarma *et al.*, 2006). A grande barreira em uma otimização como esta ainda é o tempo de avaliação da função objetivo, que é dependente de um simulador que calcula o fluxo de fluído dentro de um reservatório, tarefa essa que pode ser extremamente custosa computacionalmente. O principal objetivo neste estudo de caso é avaliar o desempenho quanto ao número de avalições frente a outros algoritmos.

4.3.2.
Simuladores de Reservatório

Na engenharia de petróleo, uma das tarefas mais importantes é a previsão da produção de hidrocarbonetos de reservatórios. Essa tarefa permite quantificar reservas, avaliar e priorizar projetos de explotação e dimensionar sistemas de produção para otimizar a produção de um reservatório. Nesse sentido, a simulação computacional de reservatórios de petróleo vem sendo utilizada há décadas e tem se tornado, cada dia mais, uma importante ferramenta para a indústria de exploração de petróleo e gás natural.

O cálculo da função objetivo (VPL) utilizada neste trabalho depende da simulação do fluxo de fluido dentro do reservatório para obtenção das curvas de produção e injeção do sistema de produção. Esta tarefa é feita por diversos softwares de simulação de reservatório comerciais.

Um simulador de reservatório é um modelo computacional capaz de gerar previsões do comportamento de um reservatório de petróleo sob diferentes condições de operação (Ertekin, Abou-Kassem e King, 2001). Este modelo tem como base a teoria de fluxo em meios porosos (SOARES e ARAÚJO, 2002), que possui um campo de aplicação bastante abrangente, sendo empregada não só na engenharia de reservatórios, mas também em diversas outras áreas da ciência e engenharia como: modelagem de aquíferos, mecânica dos solos, fluxo de contaminantes e análises de fluxo sanguíneo no interior do corpo humano.

Alguns dos modelos de fluxo mais utilizados em simulações de reservatórios são: o modelo *black-oil*, o modelo composicional, os modelos térmicos e os modelos de fluxo miscível (Mattax e Dalton, 1990).

O modelo *black oil*, que é o utilizado nesse trabalho, considera o sistema com apenas três componentes - ó1eo, água e gás - e três fases, também designadas por ó1eo, água e gás. Supõe-se que o componente óleo só existe na fase ó1eo, a componente água só existe na fase água e o componente gás pode se encontrar como gás livre no reservatório ou dissolvido no ó1eo. Este modelo considera também que a temperatura do reservatório é constante e que não há reações químicas entre os componentes. A sua utilização é recomendada para reservatórios que possuem óleos pesados e com baixa volatilidade, como os encontrados no Brasil (Cordazzo e

Jonas, 2006). Utilizou-se o simulador IMEX (CMG IMEX, 2015) neste trabalho para o cálculo do fluxo de fluido dentro do reservatório.

Por obter resultados satisfatórios em espaços de tempo aceitáveis, o modelo *black oil*, mesmo não sendo o mais completo existente, é o mais utilizado pelos simuladores comerciais de reservatórios. Os demais modelos são mais completos, entretanto, para a maioria dos casos práticos, são altamente custosos do ponto de vista computacional. Este traba1ho, por não pretender apresentar contribuição específica no que se refere à mode1agem de reservatórios, abordará apenas o modelo *black oil*.

4.3.3. Modelo de Reservatório

O experimento real visa maximizar o VPL da exploração do modelo de reservatório *benchmark* UNISIM (Avansi e Schiozer, 2015). O modelo de reservatório foi criado por um grupo de pesquisa da UNICAMP (Universidade Estadual de Campinas) (Unisim, 2015) a partir de um modelo real do Campo de Namorado da Bacia de Campos, e serve para comparar diversas metodologias de soluções futuras aplicadas a gerenciamento de reservatórios.

O reservatório é do tipo *black oil*, representado por uma malha do tipo *corner point*, com 81x58x20 células nas direções *I, J, K*, respectivamente (36739 células ativas); cada célula tem uma resolução de 100x100x8 metros. A permeabilidade e a porosidade do reservatório são heterogêneas em toda a sua extensão. O mapa da porosidade do reservatório e a configuração de poços do histórico de produção são mostrados na Figura 22 e na Figura 23, respectivamente. O histórico é um período de 1461 dias com quatro poços verticais e o horizonte total de produção é de 25 anos. A abordagem é determinística, i.e, somente um cenário geológico é levado em conta na otimização.

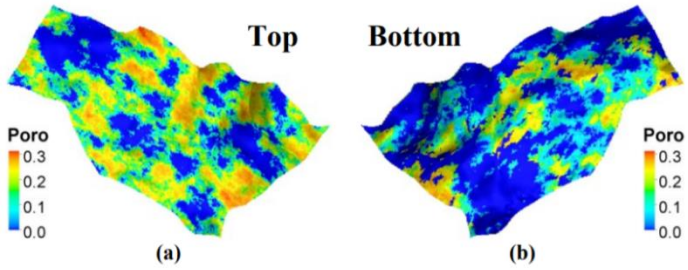


Figura 22. UNISIM-I – Modelagem petrofísica da porosidade: (a) Topo e (b) Fundo do reservatório.

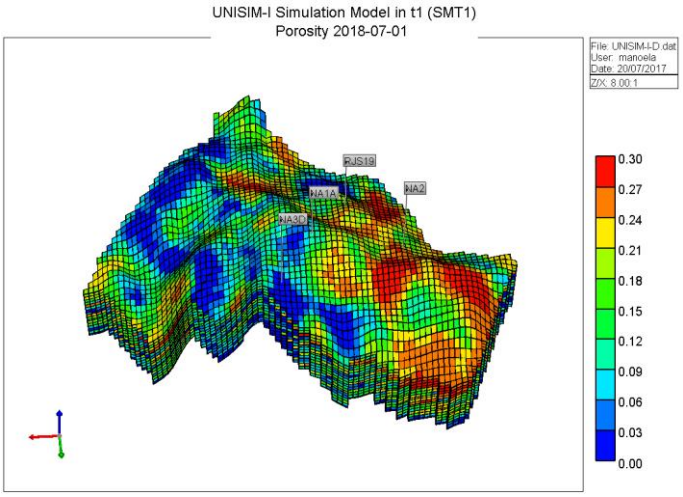


Figura 23. Poços do Histórico.

4.3.4. Modelagem do PSO+

O modelo de otimização define cada partícula como sendo composta por um conjunto válido de poços. Para representar cada poço são utilizadas seis variáveis, uma para indicar se ele está ativo (ou seja, existe fisicamente), três para representar as coordenadas *I*, *J* e *K* iniciais, uma para indicar a coordenada *K* final (as coordenadas finais *I* e *J* recebem sempre o mesmo valor das coordenadas iniciais *I* e *J*, respectivamente) e a última para representar o tipo do poço.

Assim, pode-se ver uma partícula como uma posição em um espaço multidimensional. A solução (partícula) é representada conforme a Figura 24.

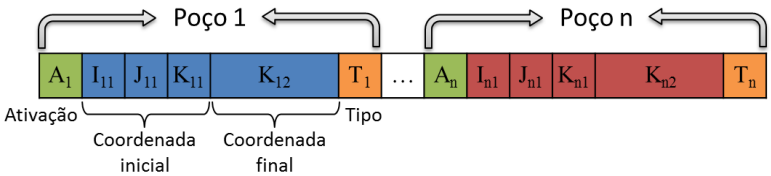


Figura 24: Representação de uma solução (ou partícula).

A representação (ou decodificação) de uma solução com dois poços – um injetor e um produtor – dentro de um reservatório é apresentada na Figura 25. O sistema de produção como um todo é apresentado na Figura 26 (onde o *riser* é uma tubulação que conecta a plataforma à *flowline* no leito marinho; a *flowline* é uma tubulação que conecta o *riser* à cabeça do poço; trecho não canhoneado do poço é o trecho que não tem contato com o reservatório; trecho canhoneado é o trecho que é de fato otimizado e faz a comunicação com o reservatório para escoamento do fluido).

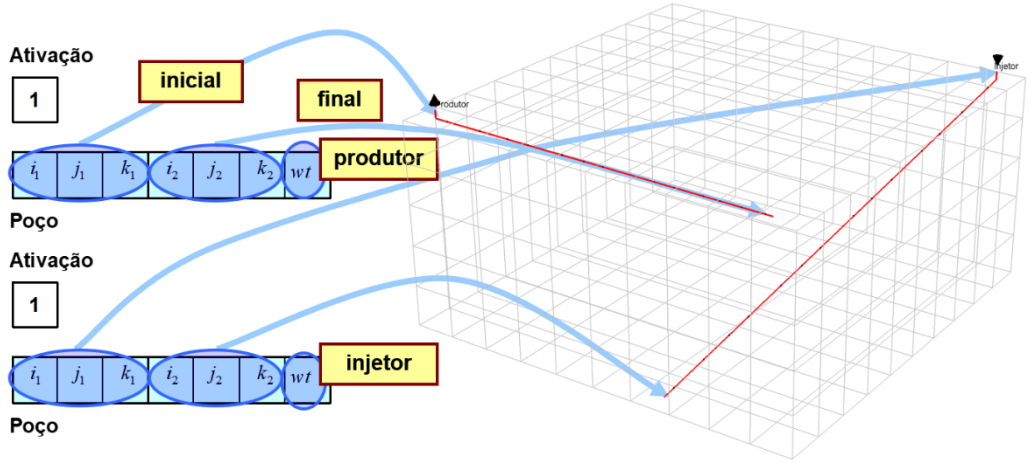


Figura 25. Exemplo de perfuração de dois poços em um reservatório.

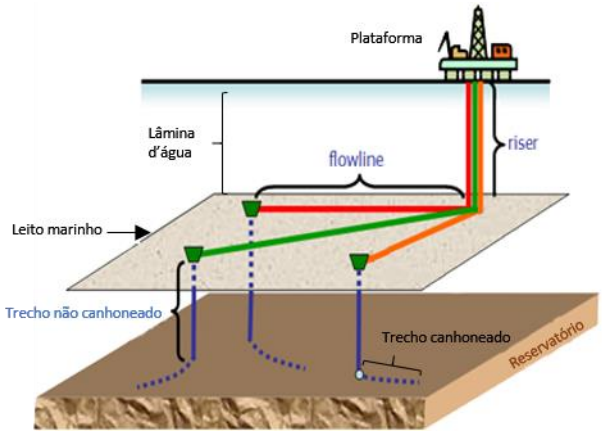


Figura 26. Representação do sistema de produção.

O número de variáveis que compõem um poço pode variar de acordo com o problema. Em uma otimização completa – onde se deseja otimizar conjuntamente todas as variáveis descritas anteriormente mais as coordenadas *I* e *J* finais –, cada poço é representado por oito variáveis. No estudo de caso apresentado, considera-

se um máximo de dez poços verticais, de forma que *I* e *J* finais não entram na representação da partícula e assumem valores iguais às coordenadas *I* e *J* iniciais, respectivamente, resultando em um total de seis variáveis por poço e sessenta variáveis no total.

O processo de otimização é discreto e as variáveis assumem valores inteiros. Algumas restrições precisam ser tratadas nesse processo:

- Limites da malha, considerando somente as células ativas do modelo.
- Número máximo de poços.
- Comprimento máximo dos poços.
- Distância mínima entre poços.
- Profundidade máxima de poços produtores (evitando assim a comunicação de poços com o reservatório em aquíferos).

Para se utilizar o PSO proposto, é necessário que ele parta de uma solução inicial e que esta seja obrigatoriamente válida, ou seja, deve obedecer a todas as restrições do problema. Desta forma, pode-se supor que existe uma distribuição de probabilidade tal que, para a geração de soluções aleatórias a partir desta distribuição, estas soluções terão alta probabilidade de serem válidas. Assim, de posse desta distribuição, a geração aleatória de soluções válidas torna-se trivial.

Nas seções que seguem é descrita em detalhes a inicialização das partículas válidas – que é diferente da inicialização das partículas para os *benchmarks* vistos até aqui – para o problema real.

4.3.4.1. Definição das Restrições do Problema de Otimização de Alocação de Poços e suas Distribuições de Probabilidade

Para iniciar o algoritmo é necessário que haja um conjunto de poços, mesmo que não sejam válidos. Para gerar a primeira amostra de poços, busca-se no banco de dados por 100 blocos. Os primeiros poços são criados de forma pontual (o que satisfaz imediatamente o tamanho máximo do poço), ou seja, o começo do poço é igual ao final do poço (são completados em somente um bloco do reservatório).

As soluções geradas aleatoriamente devem obedecer às seguintes restrições:

- Distância mínima entre poços;
- Tamanho máximo do poço;

- Linha DWOC - Os poços não podem ser gerados abaixo da linha contato óleo-água;

Escolhida a primeira amostra, calcula-se a sua verossimilhança. Esta função é a combinação de várias outras funções:

$$\mathcal{L} = F(distância) \cdot F(tamanho) \cdot F(DWOC)$$

ou seja, deve levar em consideração a distância, o tamanho do poço e a altura do contato óleo-água. Deseja-se uma função que gere poços distantes entre si, de tamanho maior possível. Todas estas funções devem ser integráveis, pois devem ser proporcionais a uma função de densidade de probabilidade.

Função Distância

A função $F(distância)$ não valoriza poços que possuam distância mínima entre eles abaixo de d_{min} (distância mínima entre poços).

$$f(d) = \begin{cases} A * tanh\big(k_1 * (x-1)\big) + 1 & , se\ x \leq 5 \\ e^{-x+log(u)+5} & se\ x > 5 \end{cases}$$

onde:

$$u = 1.9; k_1 = 10.0;$$
$$A = (u-1)/(tanh\, 4 * k_1);$$
$$d_{min} = distância\ entre\ um\ par\ de\ poços$$
$$x = \frac{distância\ entre\ um\ par\ de\ poços}{distância\ mínima\ entre\ poços}$$

Traçando a função definida acima para valores de *d* (distância) variando de 0 a 5000, com distância mínima entre poços de 500, tem-se o gráfico ilustrado na Figura 27. Pode-se observar que um par de poços com distância entre eles menor do que a distância mínima tem verossimilhança = 0,05. Ao chegar ao valor máximo, essa verossimilhança rapidamente sobe para o máximo valor 1. Esta verossimilhança mantém-se constante no valor máximo até que a distância entre os poços seja até cinco vezes maior do que a distância mínima. A partir deste valor, esta verossimilhança diminui, tendendo a zero.

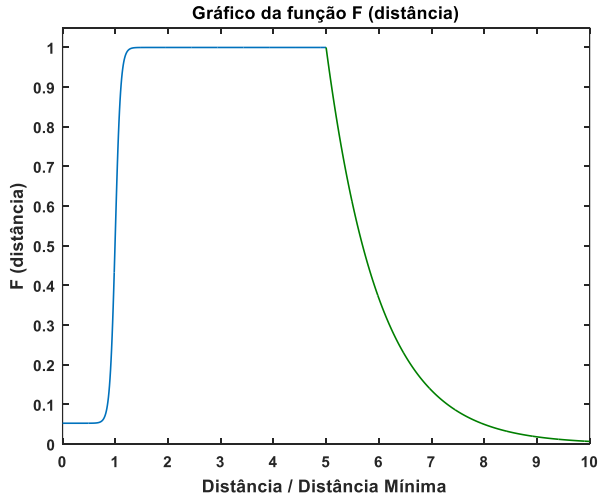


Figura 27. Gráfico da função proposta para cálculo da verossimilhança em relação à distância entre poços.

Função Tamanho

A função $F(tamanho)$ usa uma distribuição que valoriza comprimentos próximos ao tamanho máximo de poço desejado. Quanto maior o comprimento do poço acima do comprimento máximo, mais desvalorizada é a amostra, ou seja, mais próxima de zero será a verossimilhança.

$$f(l) = \begin{cases} \dfrac{1}{1 + \left(\dfrac{l}{Lmax} - 1\right)^2} & , se\ l \leq Lmax \\ e^{-k*\left(\frac{l}{Lmax} - 1\right)} & se\ l > Lmax \end{cases}$$

Onde:

$$Lmax = tamanho\ máximo\ do\ poço;\ k = 10.0;$$
$$l = comprimento\ do\ poço;$$

Valores Especiais:

$f(0) = 0.5\ (poço\ pontual)$

$f(1) = 1.0\ (poço\ no\ comprimento\ máximo)$

$f(+inf) = 0.0\ (poço\ com\ comprimento\ acima\ do\ valor\ máximo - quanto\ maior\ o\ valor, pior)$

Traçando a função definida acima para valores de l (comprimento) variando de 0 a 2000, com comprimento máximo $Lmax$ definido como 1000, o gráfico

ilustrado na Figura 28 mostra que poços pontuais (tamanho = 0) tem verossimilhança = 0.5. À medida que o comprimento do poço aumenta em direção ao tamanho máximo permitido, a verossimilhança aproxima-se do valor máximo 1. À medida que o comprimento do poço aumenta acima do valor máximo, menor é a verossimilhança daamostra.

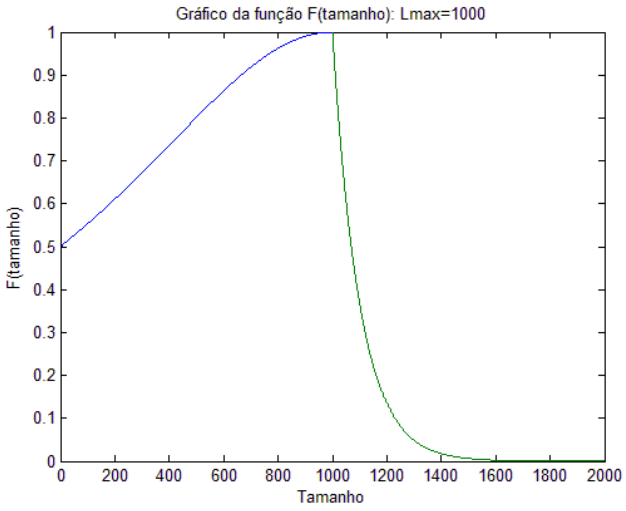


Figura 28. Gráfico da função proposta para cálculo da verossimilhança em relação ao comprimento do poço.

Função Contato Água-Óleo

A função $F(DWOC)$ verifica se a profundidade do poço gerado se encontra abaixo da linha do contato água-óleo (DWOC, do inglês *Depth of Water Oil Contact*). Caso a restrição da linha DWOC seja violada, a amostra é considerada ruim e é atribuído a ela um valor de verossimilhança próximo de zero. Caso contrário, este valor tende a 1.

$$f(x) = \frac{-arctan(x - DWOC) + \pi/2}{-arctan(-DWOC) + \pi/2}$$

Onde:

$$DWOC = profundidade\ do\ contato\ água - óleo;;$$
$$x = profundidade\ do\ poço;$$

Valores Especiais:

$f(0) = 1$

$f(inf) = 0$

Traçando a função definida acima para valores de profundidades variando de -10 a 50, com uma linha de DWOC definido em 10, tem-se o gráfico ilustrado na Figura 29. Observa-se que, à medida que a profundidade do poço ultrapassa a linha DWOC, a função de verossimilhança tende ao valor zero. Já para profundidades menores da linha DWOC, a função verossimilhança tende a 1, retornando exatamente 1 para profundidade zero.

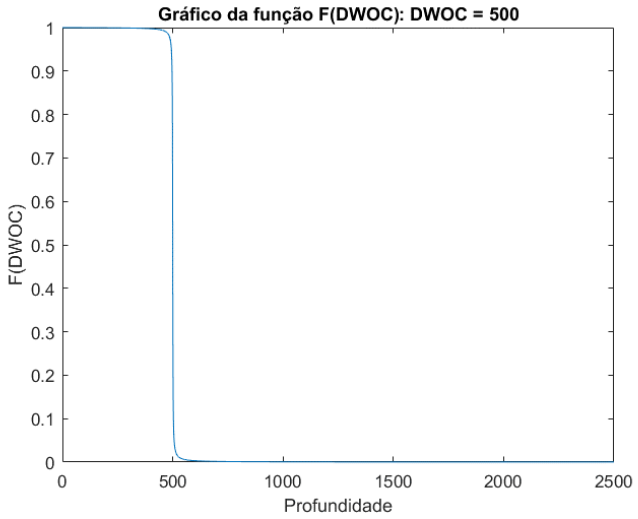


Figura 29. Gráfico da função proposta para cálculo da verossimilhança em relação ao contato óleo agua.

4.3.4.2. Metropolis-Hastings

Após a primeira amostra, o algoritmo do Metropolis-Hastings inicia-se de fato. Uma nova amostra de poços é gerada a partir da amostra inicial. A geração das amostras subsequentes a partir da amostra anterior é realizada de modo que apenas um poço e apenas uma coordenada sejam alterados (*I* inicial ou *J* inicial ou *K* inicial ou *I* final ou *J* final ou *K* final). Isso faz com que o cálculo da probabilidade condicional $p(x'|x) = p(x|x')$ (Lynch, 2007) – restrição essa que o algoritmo deve obedecer para geração de novas amostras: quando uma nova amostra é gerada, a probabilidade da transição é calculada e deve ser a mesma probabilidade da nova amostra voltar para a amostra anterior – resuma-se ao escopo das coordenadas, tornando o cálculo mais fácil e intuitivo.

A escolha de uma nova coordenada (x') a partir da coordenada atual (x) é realizada por meio de um sorteio (distribuição uniforme) entre a coordenada atual e uma faixa de valores para cima ou para baixo, chamado de δ. Ou seja, dada uma

sequência como 1, 2, 3, 4, 5, 6, 7, 8, 9 e a coordenada atual $x = 5$ e $\delta = 2$, a nova coordenada é gerada via amostragem aleatória entre $[x - \delta; x + \delta]$, ou seja, (3, 4, 5, 6, 7). Logo, a probabilidade de se chegar a qualquer uma destas coordenadas (x') quando a coordenada inicial é 5 é de $p(x'|x) = 1/5$.

Caso o resultado do sorteio seja 3, a probabilidade de chegar novamente à coordenada 5 é também 1/5 (pois as possibilidades são 1, 2, 3, 4, 5). Obedece-se, assim, à restrição $p(x'|x) = p(x|x')$.

Quando a coordenada atual é, por exemplo, $x = 2$, o intervalo do sorteio aleatório é $[0; 4]$. O problema neste caso reside na possibilidade de a coordenada 0 ser sorteada. Esta não é, obviamente, uma coordenada válida.

Para que o sorteio não gere coordenadas inválidas, opta-se por estender a série de modo a espelhá-la. Assim a nova sequência fica:

$$5 - 4 - 3 - 2 - 1 - \underbrace{\mathbf{1 - 2 - 3 - 4 - 5}}_{sequência\ original} - 5 - 4 - 3 - 2 - 1\ sequência\ estendida$$

Opta-se também por limitar o valor de δ para que não seja maior do que a série original.

$$\delta_{max} = min\left(2, \begin{cases} \dfrac{max}{2} & caso\ max\%2 = 0 \\ \dfrac{max - 1}{2} & caso\ max\%2 \neq 0 \end{cases}\right)$$

No caso em questão, o valor máximo é $\delta = 2$.

Após espelhar a série, é necessário efetuar uma conversão da série estendida para a série do sorteio, mostradas na Tabela 8.

Tabela 8. Série Espelhada e Série de Sorteio

Série Espelhada	5	4	3	2	1	1	2	3	4	5	5	4	3	2	1
Série de Sorteio	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10

A conversão é feita conforme o sistema de equações:

$$x' = \begin{cases} |x'_s| + min & , se\ x'_s < min \\ x'_s - [e + (e - 1)] & , se\ x'_s > max \end{cases}$$

onde:

$e = x'_s - max$; *max = máximo valor da série original; min = mínimo valor da série original*

Voltando ao exemplo anterior, se $x = 2$ e $x'_s = 0$, onde x'_s é o valor na série de sorteio, é necessário fazer uma conversão da série de sorteio para a série espelhada, onde $x' = |0| + 1 = 1$. A probabilidade é $p(1|2) = 2/5$, pois é possível verificar que na série espelhada o número 1 aparece duplicado dentro do intervalo, pois as possibilidades são {1;1;2;3;4}. Como o número 1 aparece duas vezes, a probabilidade de 1 ser sorteado é duas vezes maior. A probabilidade $p(2|1) = 2/5$ também, pois as possibilidades são {3;2;1;1;2} – o número 2 aparece duas vezes. O cálculo das probabilidades é dado por:

Caso 1 – Pode haver extrapolação para o lado esquerdo: $x - \delta < min$

$$p(x'|x) = \begin{cases} 0 & x + \delta < x'_s \leq max \\ \dfrac{1}{2\delta + 1} & \delta - x + min < x'_s \leq x + \delta \\ \dfrac{2}{2\delta + 1} & min \leq x'_s \leq a - x + min \end{cases}$$

Caso 2 – Pode haver extrapolação para o lado direito: $x + \delta > max$

$$p(x'|x) = \begin{cases} 0 & min < x'_s \leq x <= x - \delta \\ \dfrac{1}{2\delta + 1} & x - \delta \leq x'_s < 2.max - \delta - x + 1 \\ \dfrac{2}{2\delta + 1} & 2.max - \delta - x + 1 \leq x'_s \leq max \end{cases}$$

Caso 3 – Condições normais, não há extrapolação: $x + \delta \leq max$ e $x - \delta \geq min$

$$p(x'|x) = \begin{cases} 0 & se\ x - \delta > x'_s > x + \delta \\ \dfrac{1}{2\delta + 1} & se\ x - \delta \leq x'_s \leq x + \delta \end{cases}$$

4.3.4.3. K-Medoids

A geração aleatória descrita anteriormente pode se tornar difícil – já que se supõe que todos os poços estão ativos (ou seja, existem fisicamente) – quando o número máximo de poços a ser otimizado é muito grande. Portanto, é proposta uma modificação no fluxo normal do algoritmo para que ele se adeque melhor ao problema: já que a dificuldade de se gerarem soluções válidas incialmente está diretamente ligada às restrições impostas e à dimensão do problema, são geradas mais soluções do que a necessária, variando o número de poços ativos em cada uma

das soluções. Para determinar as soluções a serem utilizadas como inicialmente, é utilizado o algoritmo K-medoids. As seções que seguem descrevem o fluxo da geração inicial das partículas.

4.3.4.3.1. O Algoritmo

A função *PAM* (*Partitioning Around Medoids*) baseia-se na busca por *k* objetos representativos, chamado medoides, entre os objetos do conjunto de dados (Kaufman e Rousseeuw, 1987). Estes medoides são calculados de tal modo que a dissimilaridade total de todos os objetos à sua medoide mais próxima é mínima: isto é, o objetivo é encontrar um subconjunto $\{m_1, \dots, m_k\} \subset \{1, \dots, n\}$ que minimiza a função objetivo: $\sum_{i=1}^{n} min_{t=1,\dots,k} d(i, m_t)$. Cada objeto é então atribuído ao *cluster* correspondente ao medoide mais próximo, e o algoritmo segue como definido na Figura 30.

Pseudocódigo do Algoritmo PAM

1: Passo 1: *BUILD*
2: **Constrói medoides iniciais:**
- **m_1 é o objeto com menor $\sum_{i=1}^{n} d(i, m_1)$**
- **m_2 minimiza a função objetivo o máximo possível**

.
.
.

- **m_k minimiza a função objetivo o máximo possível**

3: Passo 2: *SWAP*
4: **Repete até convergir**
5: **{**
6: Considera todos os pares de objetos (*i*, *j*) com:

$$i \in \{m_1, \dots, m_k\} \; e \; j \notin \{m_1, \dots, m_k\}$$

7: Faz *swap* $i \leftrightarrow j$ que resulte em uma melhora na minimização da função objetivo
8: **}**

Figura 30. Pseudocódigo do algoritmo *PAM*

O método *pam* pode ser comparado ao bem conhecido método *k-means*, que é extensivamente estudado na literatura. No método *k-means*, o centro de cada *cluster* é definido pela média de todos os objetos do cluster. O objetivo é minimizar a soma das distâncias euclidianas ao quadrado, considerando implicitamente que cada grupo tem uma distribuição normal esférica. A função *pam* é mais robusta porque minimiza a soma das diferenças (sem potência). Além disso a *pam* não necessita de estimativas iniciais para os centros dos *clusters*.

4.3.4.3.2. Detalhes do Algoritmo

A geração aleatória descrita anteriormente pode se tornar difícil quando o número máximo de poços a ser otimizado é muito grande. Assim, na geração inicial das soluções, o algoritmo Metropolis-Hastings é implementado para gerar quatro vezes o número de partículas do enxame, variando o número de poços em cada solução. As posições relativas aos poços não gerados são iniciadas aletoriamente e definidas como inativas (ou seja, o poço não existe e não necessita ser validado em relação às restrições do problema).

As quatro chamadas ao algoritmo gerarão uma quantidade de soluções iniciais quatro vezes maior que o tamanho da população inicial do algoritmo otimizador. Portanto, neste trabalho, é proposta a utilização do algoritmo Metropolis-Hastings com apoio do algoritmo K-Medoids para que se possa determinar quais soluções foram escolhidas pelo algoritmo como solução inicial na otimização. Esta abordagem insere uma diversidade maior na população inicial, já que o passo do algoritmo Metropolis-Hastings é pequeno e atua somente sobre uma coordenada do poço a cada passo da cadeia de Markov.

As figuras a seguir ilustram uma criação de soluções iniciais válidas utilizando o algoritmo Metropolis-Hastings. A Figura 31 ilustra a inicialização utilizando somente o algoritmo e, como pode ser observado, o algoritmo foi capaz de achar somente uma alternativa válida – poços pontuais – com as restrições impostas (10 partículas com cinco poços, distância entre poços = 100 metros e tamanho máximo de poço = 1000 metros). Como somente uma alternativa válida foi encontrada, ela seria replicada para as outras nove partículas, o que geraria uma população inicial repetida, sem diversidade alguma:

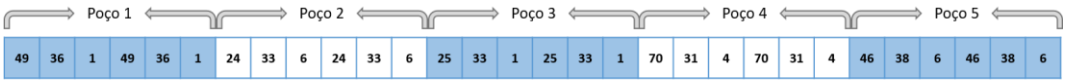


Figura 31. Soluções válidas encontradas com o algoritmo Metropolis-Hastings

A Figura 32 ilustra a inicialização utilizando o algoritmo modificado. Como se pode ver, o algoritmo foi capaz de gerar todas as dez soluções válidas diferentes, o que garante um melhor ponto de partida para o algoritmo de otimização. Os '*' na representação das soluções indicam poços desativados. As coordenadas destes não precisam respeitar quaisquer restrições, já que eles de fato não existem; então

a sua inicialização é feita de forma aleatória com distribuição uniforme, respeitando somente o domínio de cada variável (limites do *grid* do reservatório). Nos dois casos, as posições que dizem respeito ao tipo do poço são geradas de forma aleatória com distribuição uniforme.

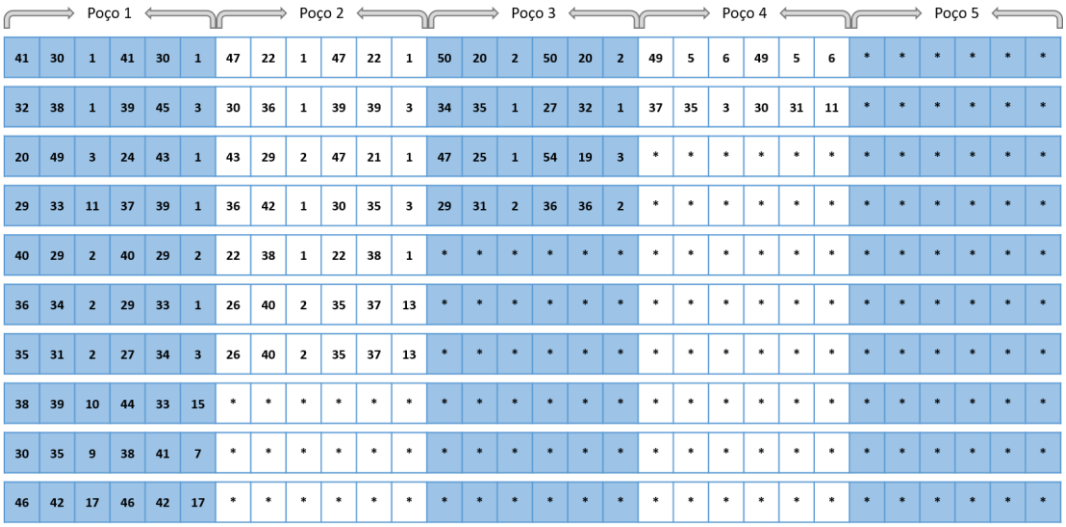


Figura 32. Soluções válidas encontradas com K-Medoids Metropolis-Hastings

4.3.5. Resultados

4.3.5.1. Parametrização do Modelo

Para o estudo de caso realizado nesta seção, foi utilizado o reservatório UNISIM descrito na seção 4.3.3, onde todos os poços criados pelo modelo são obrigatoriamente verticais e devem obedecer às restrições físicas impostas pelo problema. A Tabela 9 resume a parametrização para cada um dos algoritmos e, como se pode perceber, os experimentos foram realizados utilizando-se parâmetros tão nivelados quanto o possível para tornar a comparação entre os algoritmos a mais justa possível. A

Tabela **10** apresenta os parâmetros do cenário econômico utilizado para o cálculo da função objetivo (VPL).

Tabela 9. Parâmetros dos Algoritmos.

Parâmetros	*GA*	*PSO+*
População	30	30

Iterações	50	50
Mutação (inicial - final)	0.5-0.9	-
Cruzamento (inicial - final)	0.8-0.1	-
Steady State (inicial - final)	0.2	-
Inércia (inicial - final)	-	0.9-0.3
Rodadas Independentes	17	17
Inicialização das soluções	MH + K-Medoids	MH + K-Medoids

Tabela 10. Parâmetros do cenário econômico.

Parâmetros Econômicos	*Valor*
Preço do petróleo ($/m^3)	250
Preço do gás ($/10^{3}m^3)	50
Plataforma (Milhões de $)	500
Perfuração e completação ($/m)	15000
Custo do *Riser e Flowline* ($/m)	2000
Custo de produção de óleo ($/m^3)	40
Custo de produção de gás ($/10^{3}m^3)	2
Custo de produção de água ($/m^3)	2
Custo de injeção de água ($/m)	2
Manutenção anual do poço ($/poço)	10^6
Alíquota de impostos	34%
Taxa mínima de atratividade	9%
Royalties	10%

Todas as otimizações foram configuradas seguindo as mesmas metodologias e restrições:

1. Número máximo de poços igual a 10;
2. Os poços podem assumir dois tipos diferentes: produtor ou injetor de água;
3. O horizonte de produção é de 25 anos (01/07/2018 – 01/06/2043);
4. Todos os poços são verticais;
5. A lâmina d’água é de 160 metros;
6. A distância mínima entre poços é de 500 metros;
7. O comprimento máximo dos poços é de 500 metros;

8. O raio dos poços é de 0.0762 metros;
9. A localização da plataforma é X,Y = {354900, 7517000};
10. As restrições dos poços são as mesmas para cada tipo:
    a. Poços produtores:

       Operate STL max 1800 Cont *(Restrição de máxima taxa total de líquido (óleo + água) na superfície igual a 2000 m^3/dia)*

       Operate BHP min 190 Cont *(O poço opera normalmente até atingir a mínima pressão de fundo de poço igual a 190 KPa. Quando o poço viola essa restrição, a simulação continua, mas o poço muda para 'operando em modo de restrição violada')*

       Monitor STO min 20 Shutin *(O poço opera normalmente com a taxa de óleo na superfície maior que 20 m^3/dia. Quando o poço viola essa restrição, o poço é fechado)*

       Monitor WCUT 0.95 Shuttin *(Monitora a fração máxima de corte de água poço, que não deve passar de 95%. Quando o poço viola essa restrição, o poço é fechado)*

    b. Poços injetores de água:

       Operate BHP max 350 Cont *(Restrição de máxima pressão de fundo de poço igual a 350 kg/cm^2)*

       Operate STW max 5000 Cont *(Restrição de máxima taxa de água na superfície igual a 5000 m^3/dia)*

4.3.5.2. Análise dos Resultados

Nesta seção são apresentados os resultados relacionados ao desempenho do modelo proposto quando comparado ao sistema de otimização apresentado em (Emerick *et al.*, 2009). O sistema OCTOPUS é o resultado de uma parceria de pesquisa e desenvolvimento entre a Petrobras e o Laboratório ICA da PUC-Rio. Trata-se de um sistema desenvolvido especificamente para a solução do problema de otimização de planos de drenagem. A partir da descrição de um reservatório, o sistema busca encontrar, por meio de otimização via algoritmos genéticos, uma configuração de poços que maximize o VPL do projeto. No processo de busca pela

melhor configuração é levado em consideração a quantidade, o tipo (injetor ou produtor), a trajetória e a localização dos poços.

Os resultados obtidos após 17 rodadas de cada um dos algoritmos são apresentados na Tabela 11. Pode-se perceber que o melhor resultado de otimização foi obtido pelo PSO+ com um valor 1,53% maior do que o obtido pelo GA. Apesar de o PSO+ ter tido uma otimização com uma avaliação pior do que a pior obtida pela GA, pode-se observar que a média de avaliação das otimizações do PSO+ está 4,7% acima da média do GA. Além disso, o número de avaliações do GA para chegar ao melhor resultado foi 11,11% maior do que no PSO+. Além disso, a média de avaliações das 17 rodadas para o GA foi 37,33% maior do que para o PSO+, o que torna este um algoritmo mais indicado para a o problema de otimização de alocação de poços nesse estudo de caso proposto.

Tabela 11. Resultados do Estudo de Caso Real.

	GA	*PSO+*
Melhor	1,789,781,459.74	**1,817,233,710.84**
Pior	1,516,647,976.26	**1,463,216,576.05**
Média	1,630,805,498.58	**1,707,557,525.71**
Desvio Padrão	55,956,714.48	**79,102,476.88**
Avaliações do Melhor	1500	**1350**
Média de Avaliações	1431	**1042**

A evolução do processo de otimização para os melhores resultados de ambos os algoritmos pode ser verificada e comparada na Figura 33 e na Figura 34. As curvas de evolução fornecem as seguintes métricas de avaliação em relação ao VPL:

- A curva *Melhor* apresenta a melhor solução encontrada em cada geração do algoritmo genético.
- A curva *Média* é obtida a partir da média das avaliações de todas as soluções da iteração corrente. Esta curva permite verificar a rápida obtenção de boas soluções e também permite visualizar o grau de convergência das soluções da população.
- A curva *Offline* apresenta o valor médio das avaliações das melhores soluções encontradas a cada iteração da evolução. Esta curva mostra a qualidade do algoritmo para encontrar soluções boas desde o início.

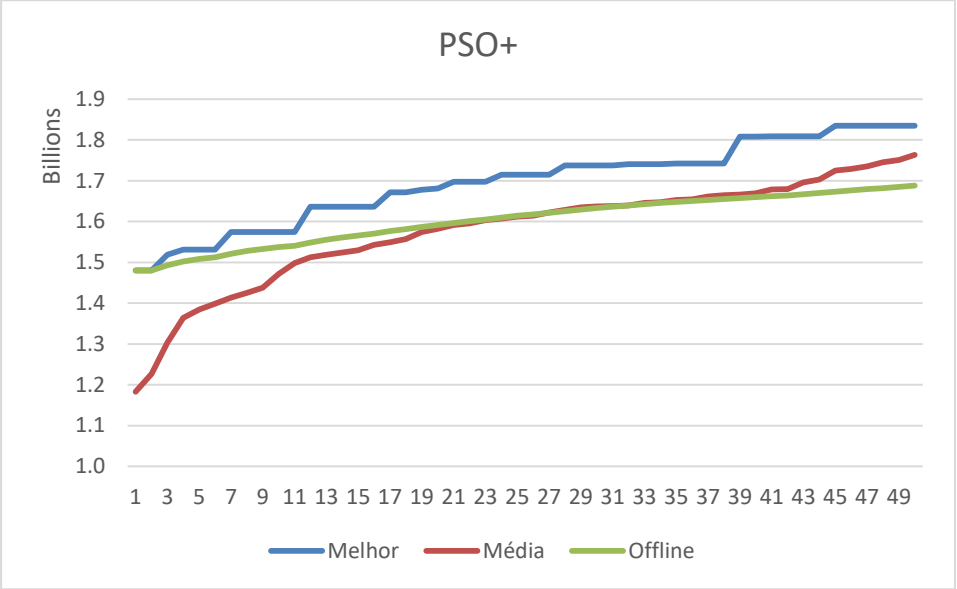


Figura 33. Evolução da otimização para o melhor resultado do PSO+.

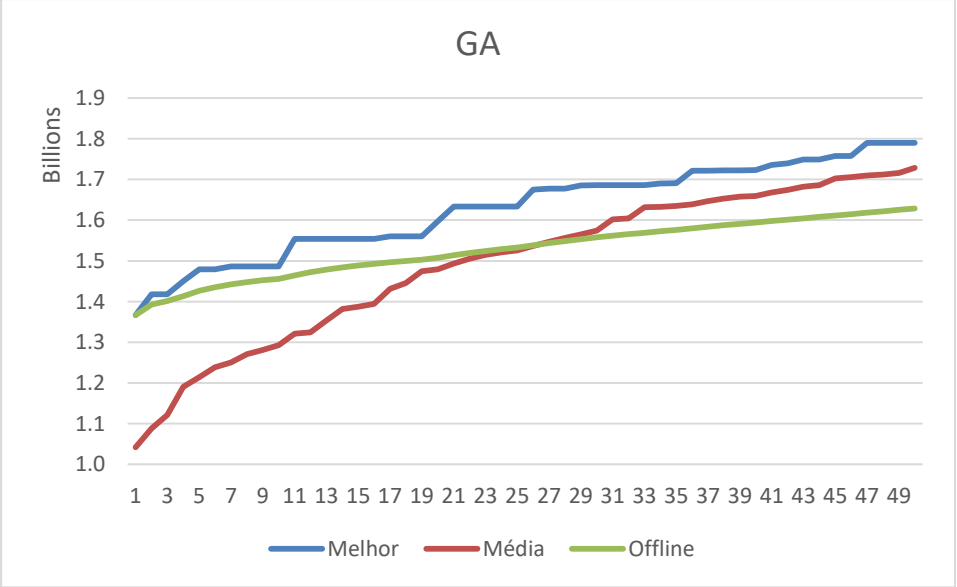


Figura 34. Evolução da otimização para o melhor resultado do GA.

O teste estatístico *Wilcoxon Signed Rank* (Levine, Ramsey e Smidt, 2001) também foi utilizado para comparar as duas amostras geradas. O teste de Wilcoxon é um teste não paramétrico (Wilcoxon, 1945) e foi utilizado para determinar se a diferença entre os resultados obtidos pelo PSO+ e pelo GA é significativa. Os melhores desempenhos obtidos pelo PSO+ e pelo GA no final de cada rodada independente são considerados para este teste. A hipótese nula é que não há

diferença entre os resultados obtidos. O valor *p-value* calculado é de 0,004507, ou seja, um valor menor do que 0.05 (intervalo de confiança), confirmando que a hipótese nula pode ser rejeitada e, consequentemente, os diferentes resultados obtidos pelos dois modelos podem ser considerados estatisticamente significativos. Deve-se observar, a partir da Figura 35, que os desempenhos obtidos pelo PSO+ em cada rodada são maiores (melhor) do que os do GA. Além disso, também pode-se perceber, analisando o *boxplot*, que o melhor resultado obtido pelo GA (pior que o melhor obtido pelo PSO+) é um *outlier*. Assim, pode-se concluir que o PSO+ apresenta um melhor desempenho que o GA para o estudo de caso proposto.

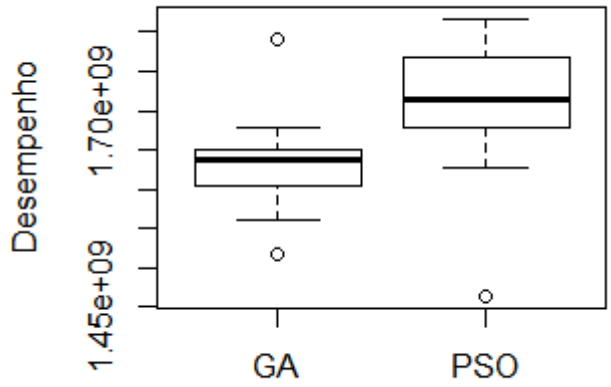


Figura 35. *Boxplot* dos desempenhos das 17 rodadas independentes de cada um dos modelos de otimização.

4.4. Considerações de Implementação

Todo o código utilizado para este trabalho foi desenvolvido na linguagem C# e no MS .Net Framework 4.5. Os sistemas auxiliares utilizados foram: o simulador IMEX e gerador de tabelas *Results Report* da CMG; e o sistema de distribuição de tarefas usado é o Torque (Staples e Garrick, 2006) (do inglês *Terascale Open-source Resource and QUEue Manager*), um gerenciador de recursos distribuídos que fornece controle sobre trabalhos em lote e nós de computação distribuídos.

5.
Considerações Finais

5.1.
Conclusões

Este trabalho apresentou um novo algoritmo de otimização por enxame de partículas para otimização com restrições lineares e não lineares. Foram propostas a inserção de seis novos módulos ao PSO original, de forma a garantir capacidades como: devida exploração da borda do espaço de busca válida, garantia de geração de soluções válidas, múltiplos enxames para separar devidamente os papéis de cada uma das partículas, garantia de diversidade da população, melhor exploração do espaço de busca e minimização do problema de convergência prematura.

Ao utilizar o operador de redirecionamento aritmético para ser usado com o PSO, permite-se que as partículas inválidas facilmente se redirecionem para uma região válida com um baixo custo computacional. Esta abordagem também permite que o algoritmo mantenha uma população inteira de partículas válidas em qualquer momento da otimização. O conceito de pontos de apoio funciona bem quando as restrições de otimização são ativas na solução ótima, permitindo que o enxame, além de manter alguma diversidade nas soluções e melhorar a cobertura do espaço de busca válido, possa alcançar mais facilmente a fronteira do espaço de busca válido. Os pontos de apoio auxiliam o algoritmo em alguns problemas de *benchmark* com restrições ativas no ótimo.

A heurística de inicialização de partículas permite que o estágio inicial da otimização não seja tão custoso do ponto de vista computacional quando o problema apresenta restrições que dificultem a procura por soluções válidas. O uso e modificação do algoritmo *Metropolis-Hastings* para a inicialização também de soluções válidas torna o custo computacional ainda menor – especialmente quando a amostragem aleatória deve ser feita a partir de uma distribuição de probabilidade que não pode ser gerada (ou é de difícil criação) de modo direto – e garante que a população inicial seja iniciada sem problemas, além de gerar uma população mais diversa inicialmente.

Além disso, a topologia de vizinhança, denominada vizinhança de agrupamento aleatório coordenado, é usado para minimizar o problema de convergência prematura da otimização.

Os resultados obtidos com os cinco estudos de caso com restrições de domínio apontam que as funcionalidades propostas para o modelo influenciam diretamente no resultado de otimização e na quantidade de avaliações até a convergência, por isso, todos os módulos foram utilizados para os estudos de caso seguintes.

Os resultados obtidos com alguns estudos de caso de otimização numérica com restrições permitem avaliar positivamente o PSO+, que apresentou resultados de otimização e velocidade de convergência, em sua maioria, melhores que três modelos de algoritmos genéticos (RGenIII, GenV e RGenV), quatro implementações PSO diferentes que lidam com problemas com restrições (PSO_Eb, Micro-PSO, PSO_CO e COPSO), além de três outros algoritmos baseados em colônia artificial de abelhas, algoritmo meta-heurístico de vaga-lume e um algoritmo baseado no comportamento social de aranhas (eABC, FA e SSO-C, respectivamente).

Para o estudo de caso real, o PSO+ mostrou ser um algoritmo mais adequado para o problema de alocação de poços no modelo de reservatório proposto. Além da obtenção de uma alternativa de poços com um VPL ligeiramente melhor que o melhor VPL obtido com GA, o número de avalições necessárias para a convergência foi significativamente menor do que no GA.

Em geral, o PSO+ se mostrou um método eficiente e geral para resolver problemas de otimização com restrições.

5.2. Trabalhos Futuros

Um problema real de otimização não linear com restrições não lineares é a otimização de malhas de drenagens sob incerteza geológica. Esse tipo de otimização, considerando incerteza, é geralmente custoso do ponto de vista computacional em virtude da quantidade de simulações necessárias.

O tratamento eficiente da incerteza, especialmente a de natureza geológica, representa um desafio para a aplicação prática dos algoritmos de otimização. A geologia de subsuperfície é altamente incerta e, para se representar o alto grau de

incerteza na geologia do reservatório, muitas realizações geológicas devem ser consideradas. A aplicação direta da abordagem em que todas as realizações são consideradas em todas as iterações leva a otimizações computacionalmente custosas, particularmente quando o número de realizações é grande.

A otimização de alocação de poços sob incerteza geológica tem sido considerada por inúmeros pesquisadores, assim como os estudos que abordam somente o problema de localização de poços (Artus *et al.*, 2006; Güyagüler e Horne, 2004; Onwunalu e Durlofsky, 2010; Özdogan e Horne, 2006). Nestas aplicações, a otimização foi realizada com um número fixo e reduzido de realizações, porque, como indicado anteriormente, quanto maior o número de realizações consideradas, maior o custo computacional da otimização, a qual pode, inclusive, se tornar proibitiva.

Entender e capturar a incerteza dos reservatórios são as chaves para prever seu desempenho e tomar decisões operacionais. Práticas convencionais da indústria com um ou três modelos (P10, P50, P90) têm pouca habilidade para descrever toda a complexidade da incerteza de subsuperfície, o que resulta em um baixo desempenho nas previsões de produção. Em (Wang *et al.*, 2012; Yeh *et al.*, 2014), os autores propõem novas abordagens para lidar com o problema de otimização de alocação de poços sob incerteza geológica de forma a minimizar os altos custos de simulação.

Um possível trabalho futuro seria a proposição de um método para tratamento de incertezas geológicas de forma a não descartar grande parte das realizações geológicas e de não perder informação relevante à alocação ótima dos poços. Outro possível aprimoramento no processo de otimização para o estudo de caso real seria a representação de posicionamento das partículas por ordem: um PSO sem posicionamento por ordem pode levar a muitas soluções ótimas diferentes que, na verdade, representam a mesma solução física. Além disso, a redundância no domínio do problema faz com que o espaço de busca fique desnecessariamente grande, de modo que o algoritmo tende a convergir mais lentamente para uma solução relevante.

Referências bibliográficas

ABDESSAMIA, F.; TAI, Y.; ZHANG, W. Z.; SHAFIQ, M. An Improved Particle Swarm Optimization for Energy-Efficiency Virtual Machine Placement. International Conference on Cloud Computing Research and Innovation (ICCCRI). **Anais**...IEEE, abr. 2017 Disponível em: <http://ieeexplore.ieee.org/document/7999634/>. Acesso em: 4 ago. 2017

AGUIRRE, A. H.; ZAVALA, A. E. M.; VILLA DIHARCE, E.; BOTELLO RIONDA, S.; MUÑOZ, A. E.; VILLA DIHARCE, Z. E. COPSO: Constrained Optimization Via PSO Algorithm. *In*: **Constraint-Handling in Evolutionary Optimization**. [s.l.] Springer, 2009. .

AHMAD, R.; BABAEIZADEH, S. An Efficient Artificial Bee Colony Algorithm for Constrained Optimization Problems. **Journal of Engineering and Applied Sciences**, v. 9, p. 405–413, 2014.

ARTUS, V.; DURLOFSKY, L. J.; ONWUNALU, J.; AZIZ, K. Optimization of nonconventional wells under uncertainty using statistical proxies. **Computational Geosciences**, v. 10, n. 4, p. 389–404, 30 nov. 2006.

AVANSI, G. D.; SCHIOZER, D. J. UNISIM-I: Synthetic Model for Reservoir Development and Management Applications. **International Journal of Modeling and Simulation for the Petroleum Industry**, v. 9, n. 1, 2015.

BANGERTH, W.; KLIE, H.; WHEELER, M. F.; STOFFA, P. L.; SEN, M. K. On optimization algorithms for the reservoir oil well placement problem. **Computational Geosciences**, v. 10, n. 3, p. 303–319, 17 set. 2006.

BERGH, F. VAN DEN; ENGELBRECHT, A. P. **A New Locally Convergent Particle Swarm Optimiser**IEEE International Conference on Systems, Man and Cybernetics. **Anais**...IEEE, 2002 Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1176018>. Acesso em: 6 jul. 2016

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

CHANG, W.-D.; SHIH, S.-P. PID controller design of nonlinear systems using an

improved particle swarm optimization approach. **Communications in Nonlinear Science and Numerical Simulation**, v. 15, n. 11, p. 3632–3639, 2010.

CHENG, G.; AN, Y.; WANG, Z.; ZHU, K. **Oil Well Placement Optimization Using Niche Particle Swarm Optimization**2012 Eighth International Conference on Computational Intelligence and Security. **Anais**...IEEE, nov. 2012 Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6405867>. Acesso em: 13 jun. 2016

CLERC, M. **Particle Swarm Optimization**. [s.l.] iSTE, London, England, 2006.

CMG IMEX. **IMEX User Guide - Three-Phase, Black-Oil Reservoir Simulator**Calgary, Alberta, Canada: Computer Modelling Group LTD, 2015.

COELLO, C. A. C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. **Computer Methods in Applied Mechanics and Engineering**, v. 191, n. 11, p. 1245–1287, 2002.

COELLO COELLO, C. A.; A., C. **Constraint-handling techniques used with evolutionary algorithms**Proceedings of the 12th annual conference comp on Genetic and evolutionary computation - GECCO '10. **Anais**...New York, New York, USA: ACM Press, 2010 Disponível em: <http://portal.acm.org/citation.cfm?doid=1830761.1830910>. Acesso em: 11 jul. 2017

CORDAZZO, J.; JONAS. Simulação de reservatórios de petróleo utilizando o método EbFVM e multigrid algébrico. 2006.

CORNE, D.; GLOVER, F.; DORIGO, M. **An Introduction to Differential Evolution, New Ideas in Optimization**. London, U.K.: McGraw-Hill, 1999.

CUEVAS, E.; CIENFUEGOS, M. A new algorithm inspired in the behavior of the social-spider for constrained optimization. **Expert Systems with Applications**, v. 41, n. 2, p. 412–425, fev. 2014.

DEB, K. An efficient constraint handling method for genetic algorithms. **Computer Methods in Applied Mechanics and Engineering**, v. 186, n. 2–4, p. 311–338, jun. 2000.

DERRAC, J.; GARCÍA, S.; MOLINA, D.; HERRERA, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, v. 1, n. 1, p. 3–18, 1 mar. 2011.

DESHPANDE, A. M.; PHATNANI, G. M.; KULKARNI, A. J. **Constraint**

handling in Firefly Algorithm2013 IEEE International Conference on Cybernetics (CYBCO). **Anais**...IEEE, jun. 2013 Disponível em: <http://ieeexplore.ieee.org/document/6617447/>. Acesso em: 27 jul. 2017

DOMINGUEZ-ISIDRO, S.; MEZURA-MONTES, E.; LEGUIZAMON, G. **Memetic differential evolution for constrained numerical optimization problems**2013 IEEE Congress on Evolutionary Computation. **Anais**...IEEE, jun. 2013 Disponível em: <http://ieeexplore.ieee.org/document/6557934/>. Acesso em: 25 maio. 2017

EBERHART, R. C.; SHI, Y. **Comparing inertia weights and constriction factors in particle swarm optimization**Proceedings of the 2000 Congress on Evolutionary Computation. **Anais**...La Jolla, CA: 2000 Disponível em: <http://ieeexplore.ieee.org/document/870279/>. Acesso em: 29 maio. 2017

EMERICK, A. A.; SILVA, E.; MESSER, B.; ALMEIDA, L. F.; SZWARCMAN, D.; PACHECO, M. A. P.; VELLASCO, M. M. B. R. Well Placement Optimization Using a Genetic Algorithm with Nonlinear Constraints. **SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA**, n. SPE118808, 2009.

ENGELBRECHT, A. P. **Fundamentals of computational swarm intelligence**. [s.l.] Wiley, West Sussex, England, 2005.

ENGELBRECHT, A. P. **Computational Intelligence: An Introduction**. 2nd ed. ed. [s.l.] Wiley Publishing, 2007.

ERTEKIN, T.; ABOU-KASSEM, J. H. (JAMAL H.; KING, G. R. **Basic applied reservoir simulation**. [s.l.] Society of Petroleum Engineers, 2001.

FIRMINO, P. R. A. **Método Adaptativo de Markov Chain Monte Carlo para Manipulação de Modelos Bayesianos**. [s.l: s.n.].

FUENTES CABRERA, J.; COELLO COELLO, C. Handling constraints in particle swarm optimization using a small population size. **MICAI 2007: Advances in Artificial Intelligence**, v. 4827, p. 41–51, 2007.

GÜYAGÜLER, B.; HORNE, R. N. Uncertainty Assessment of Well-Placement Optimization. **SPE Reservoir Evaluation & Engineering**, v. 7, n. 1, p. 24–32, 2004.

HASTINGS, W. K. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. **Biometrika**, v. 57, n. 1, p. 97–109, 1970.

HE, Q.; WANG, L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. **Applied Mathematics and Computation**, v.

186, n. 2, p. 1407–1422, 2007.

HOANG, T. T.; CHO, M.-Y.; ALAM, M. N.; VU, Q. T. A novel differential particle swarm optimization for parameter selection of support vector machines for monitoring metal-oxide surge arrester conditions. **Swarm and Evolutionary Computation**, jul. 2017.

HOU, Y.; WU, N.; ZHOU, M.; LI, Z. Pareto-Optimization for Scheduling of Crude Oil Operations in Refinery via Genetic Algorithm. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 47, n. 3, p. 517–530, mar. 2017.

HU, X.; EBERHART, R. Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. **Optimization**, v. 2, n. 1, p. 1677–1681, 2002.

JAN, M. A.; TAIRAN, N.; KHANUM, R. A. **Threshold Based Dynamic and Adaptive Penalty Functions for Constrained Multiobjective Optimization**2013 1st International Conference on Artificial Intelligence, Modelling and Simulation. **Anais**...IEEE, dez. 2013 Disponível em: <http://ieeexplore.ieee.org/document/6959893/>. Acesso em: 11 jul. 2017

JANSON, S.; MIDDENDORF, M. A hierarchical particle swarm optimizer and its adaptive variant. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 35, n. 6, p. 1272–1282, dez. 2005.

JIE LI; SAVA, A.; XIAOLAN XIE. Simulation-Based Discrete Optimization of Stochastic Discrete Event Systems Subject to Non Closed-Form Constraints. **IEEE Transactions on Automatic Control**, v. 54, n. 12, p. 2900–2904, dez. 2009.

KARABOGA, D. AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMIZATION. **Computer Engineering Department, Erciyes University, Kayseri, Turkey. Technical Report TR06**, 2005.

KATO, K.; SAKAWA, M.; KATAGIRI, H. Revision of a Floating-Point Genetic Algorithm GENOCOP V for Nonlinear Programming Problems. **The Open Cybernetics & Systemics Journal** , v. 2, n. 1, p. 24–29, 2008.

KAUFMAN, L.; ROUSSEEUW, P. **Clustering by means of medoids - Statistical data analysis based on the L_1-norm and related methods**. North-Holland, Amsterdam: North-Holland, 1987.

KAUR, A.; CHEEMA, S. S. **A hybrid multi-agent based particle swarm optimization for telemedicine system for neurological disease**2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE). **Anais**...IEEE, dez. 2016 Disponível em:

<http://ieeexplore.ieee.org/document/7939527/>. Acesso em: 4 ago. 2017

KENNEDY, J.; EBERHART, R. Particle swarm optimization. **Neural Networks, 1995. Proceedings., IEEE International Conference on**, v. 4, p. 1942–1948 vol.4, 1995.

KENNEDY, J.; MENDES, R. **Population structure and particle swarm performance**Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). **Anais**...IEEE, 2002 Disponível em: <http://ieeexplore.ieee.org/document/1004493/>. Acesso em: 3 jul. 2017

KOHLER, M.; FORERO, L.; VELLASCO, M.; TANSCHEIT, R.; PACHECO, M. A. **PSO+: A nonlinear constraints-handling particle swarm optimization**2016 IEEE Congress on Evolutionary Computation, CEC 2016. **Anais**...2016

KOZIEL, S., MICHALEWICZ, Z. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. v. 7, n. 2, p. 19–44, 1999.

KRAMER, O.; OLIVER. A Review of Constraint-Handling Techniques for Evolution Strategies. **Applied Computational Intelligence and Soft Computing**, v. 2010, p. 1–11, 18 abr. 2010.

KRAMER, O.; SCHWEFEL, H.-P. On three new approaches to handle constraints within evolution strategies. **Natural Computing**, v. 5, n. 4, p. 363–385, 2 nov. 2006.

KUMAR, R.; ANAND, S.; SYDULU, M. A new multi agent based PSO approaches for optimal power flows with security constraints using different cost functions. **2011 2nd International Conference on Intelligent Agent & Multi-Agent Systems**, p. 67–72, 2011.

LEVINE, D. M.; RAMSEY, P. P.; SMIDT, R. K. **Applied statistics for engineers and scientists**. [s.l.] Prentice Hall, 2001.

LI, J.-P.; WANG, Y.; YANG, S.; CAI, Z. **A comparative study of constraint-handling techniques in evolutionary constrained multiobjective optimization**2016 IEEE Congress on Evolutionary Computation (CEC). **Anais**...IEEE, jul. 2016 Disponível em: <http://ieeexplore.ieee.org/document/7744320/>. Acesso em: 11 jul. 2017

LI, X.; TIAN, P.; KONG, M. A Novel Particle Swarm Optimization for Constrained Optimization Problems. **AI 2005: Advances in Artificial Intelligence, 18th Australian Joint Conference on Artificial Intelligence**, p. 1305–1310, 2005.

LIANG, J. J.; QU, B. Y.; SUGANTHAN, P. N.; NIU, B. **Dynamic Multi-Swarm**

Particle Swarm Optimization for Multi-objective optimization problems2012 IEEE Congress on Evolutionary Computation. **Anais**...IEEE, jun. 2012 Disponível em: <http://ieeexplore.ieee.org/document/6256416/>. Acesso em: 5 jun. 2017

LIANG, J.; RUNARSSON, T. P.; MEZURA-MONTES, E.; CLERC, M.; SUGANTHAN, P.; COELLO, C. A. C.; DEB, K. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. v. 41, 2006.

LIU, C.-A. New Multiobjective PSO Algorithm for Nonlinear Constrained Programming Problems. *In*: **Advances in Cognitive Neurodynamics ICCN 2007**. Dordrecht: Springer Netherlands, 2008. p. 955–962.

LU, L.; LUO, Q.; LIU, J.; LONG, C. An improved particle swarm optimization algorithm. **IEEE International Conference on Granular Computing**, p. 486–490, 2008.

LYNCH, S. M. **Introduction to Applied Bayesian Statistics and Estimation for Social Scientists**. New York, NY: Springer New York, 2007.

MATTAX, C. C.; DALTON, R. L. **Reservoir simulation**. [s.l.] Henry L. Doherty Memorial Fund of AIME, Society of Petroleum Engineers, 1990.

MICHALEWICZ, Z. **A Survey of Constraint Handling Techniques in Evolutionary Computation Methods**Proceedings of the 4th Annual Conference on Evolutionary Programming. **Anais**...1995

MICHALEWICZ, Z.; JANIKOW, C. Z. GENOCOP: a genetic algorithm for numerical optimization problems with linear constraints. **Communications of the ACM**, v. 39, n. 12es, p. 175–es, 1 dez. 1996.

MICHALEWICZ, Z.; NAZHIYATH, G. **Genocop III: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints**Proceedings of 1995 IEEE International Conference on Evolutionary Computation. **Anais**...IEEE, 1995 Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=487460>. Acesso em: 19 jan. 2016

MICHALEWICZ, Z.; SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. **Evol. Comput**, v. vol, p. 4no1pp1-32, 1996.

NARIÑO, G. A. R. **Otimização de Risers em Catenária com Amortecedores Hidrodinâmicos**. [s.l: s.n.].

ONWUNALU, J. E.; DURLOFSKY, L. J. Application of a particle swarm optimization algorithm for determining optimum well location and type. **Computational Geosciences**, v. 14, p. 183–198, 2010.

ÖZDOGAN, U.; HORNE, R. N. Optimization of Well Placement Under Time-Dependent Uncertainty. **SPE Reservoir Evaluation & Engineering**, v. 9, n. 2, p. 135–145, 1 abr. 2006.

PAREDIS, J. Co-evolutionary constraint satisfaction. *In*: [s.l.] Springer, Berlin, Heidelberg, 1994. p. 46–55.

PARK, C.; KIM, S.-H. Penalty Function with Memory for Discrete Optimization via Simulation with Stochastic Constraints. **Operations Research**, v. 63, n. 5, p. 1195–1212, out. 2015.

PEIMANKAR, A.; WEDDELL, S. J.; JALAL, T.; LAPTHORN, A. C. Evolutionary multi-objective fault diagnosis of power transformers. **Swarm and Evolutionary Computation**, abr. 2017.

POLI, R.; KENNEDY, J.; BLACKWELL, T. Particle swarm optimization. **Swarm Intelligence**, v. 1, n. 1, p. 33–57, 17 out. 2007.

QOLOMANY, B.; MAABREH, M.; AL-FUQAHA, A.; GUPTA, A.; BENHADDOU, D. **Parameters optimization of deep learning models using Particle swarm optimization**2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC). **Anais**...IEEE, jun. 2017 Disponível em: <http://ieeexplore.ieee.org/document/7986470/>. Acesso em: 4 ago. 2017

RAITOHARJU, J.; SAMIEE, K.; KIRANYAZ, S.; GABBOUJ, M. Particle swarm clustering fitness evaluation with computational centroids. **Swarm and Evolutionary Computation**, v. 34, p. 103–118, jun. 2017.

RAQUEL, C. R.; NAVAL, P. C. **An effective use of crowding distance in multiobjective particle swarm optimization**Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05. **Anais**...New York, New York, USA: ACM Press, 2005 Disponível em: <http://portal.acm.org/citation.cfm?doid=1068009.1068047>. Acesso em: 6 jul. 2016

REY, A.; ZMEUREANU, R. Micro-time variant multi-objective particle swarm optimization (micro-TVMOPSO) of a solar thermal combisystem. **Swarm and Evolutionary Computation**, abr. 2017.

ROBINSON, J.; RAHMAT-SAMII, Y. Particle swarm optimization in

electromagnetics. **Antennas and Propagation, IEEE Transactions on**, v. 52, n. 2, p. 397–407, 2004.

SAHA, C.; DAS, S.; PAL, K.; MUKHERJEE, S. A Fuzzy Rule-Based Penalty Function Approach for Constrained Evolutionary Optimization. **IEEE Transactions on Cybernetics**, v. 46, n. 12, p. 2953–2965, dez. 2016.

SARMA, P.; DURLOFSKY, L.; AZIZ, K.; CHEN, W. Efficient real-time reservoir management using adjoint-based optimal control and model updating. **Computational Geosciences**, v. 10, n. 1, p. 3–36, 2006.

SCHOENAUER, M.; MICHALEWICZ, Z. **Evolutionary Computation at the Edge of Feasibility**Proceedings of the 4th Conference on Parallel Problem Solving from Nature (PPSN '96). **Anais**...Berlin, Germany: 1996 Disponível em: <https://cs.adelaide.edu.au/~zbyszek/Papers/p26.pdf>

SCHOENAUER, M.; XANTHAKIS, S. Constrained GA optimization. **Urbana Champaign**, 1993.

SHI, Y.; EBERHART, R. **A Modified Particle Swarm Optimizer**IEEE World Congress on Computational Intelligence. **Anais**...Anchorage, AK: 1998

SOARES, A. A. M.; ARAÚJO, É. R. **Simulação de Reservatórios de Petróleo em Arquiteturas Paralelas com Memória Distribuida**. [s.l.] UFPE, Recife, 2002.

STAPLES, G.; GARRICK. **TORQUE---TORQUE resource manager**Proceedings of the 2006 ACM/IEEE conference on Supercomputing - SC '06. **Anais**...New York, New York, USA: ACM Press, 2006 Disponível em: <http://portal.acm.org/citation.cfm?doid=1188455.1188464>. Acesso em: 14 ago. 2017

SUGANTHAN, P. N. **Particle swarm optimiser with neighbourhood operator**Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). **Anais**...IEEE, 1999Disponível em: <http://ieeexplore.ieee.org/document/785514/>. Acesso em: 5 jun. 2017

ULLAH, A. S. S. M. B.; ELFEKY, E. Z.; CORNFORTH, D.; ESSAM, D. L.; SARKER, R. **Improved evolutionary algorithms for solving constrained optimization problems with tiny feasible space**2008 IEEE International Conference on Systems, Man and Cybernetics. **Anais**...IEEE, out. 2008 Disponível em: <http://ieeexplore.ieee.org/document/4811486/>. Acesso em: 5 jun. 2017

UNISIM. **Grupo de Pesquisa na área de simulação numérica e gerenciamento de reservatórios de petróleo**. Disponível em:

<https://www.unisim.cepetro.unicamp.br/br/>. Acesso em: 10 nov. 2016.

VEGA-ALVARADO, E.; PORTILLA-FLORES, E. A.; CALVA-YANEZ, M. B.; SEPULVEDA-CERVANTES, G.; APONTE-RODRIGUEZ, J. A.; SANTIAGO-VALENTIN, E.; RUEDA-MELENDEZ, J. A. Hybrid Metaheuristic for Designing an End Effector as a Constrained Optimization Problem. **IEEE Access**, p. 1–1, 2017.

WANG, H.; ECHEVERRÍA-CIAURRI, D.; DURLOFSKY, L.; COMINELLI, A. Optimal Well Placement Under Uncertainty Using a Retrospective Optimization Framework. **Society of Petroleum Engineers**, v. 17, n. 1, p. 112–121, 2012.

WANG, X.; CHOI, T.-M.; LIU, H.; YUE, X. Novel Ant Colony Optimization Methods for Simplifying Solution Construction in Vehicle Routing Problems. **IEEE Transactions on Intelligent Transportation Systems**, v. 17, n. 11, p. 3132–3141, nov. 2016.

WHITLEY, D.; GORDON, V. S.; MATHIAS, K. Lamarckian evolution, the Baldwin effect and function optimization. *In*: [s.l.] Springer, Berlin, Heidelberg, 1994. p. 5–15.

WILCOXON, F. Individual Comparisons by Ranking Methods. **Biometrics Bulletin**, v. 1, n. 6, p. 80–83, 1945.

XU, Y.; LEI, B.; SUN, S. Three particle swarm algorithms to improve coverage of camera networks with mobile nodes. **Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference**, n. 1, p. 816–820, 2010.

YEH, T.-H.; JIMENEZ, E.; ESSEN, G. VAN; CHEN, C.; JIN, L.; GIRARDI, A.; GELDERBLOM, P.; HORESH, L.; CONN, A. R. **Reservoir Uncertainty Quantification Using Probabilistic History Matching Workflow**SPE Annual Technical Conference and Exhibition. **Anais**...Society of Petroleum Engineers, 27 out. 2014Disponível em: <http://www.onepetro.org/doi/10.2118/170893-MS>. Acesso em: 6 jul. 2016

YU, E.; FEI, Q.; MA, H.; GENG, Q. **Improving constraint handling for multiobjective particle swarm optimization**Proceedings of the 33rd Chinese Control Conference. **Anais**...IEEE, jul. 2014Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6896448>. Acesso em: 11 jul. 2017

Apêndice A – Funções *Benchmark* para Otimização com Restrições Lineares e Não Lineares

a) Função G1

Minimizar:

$$G1(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

Sujeito às seguintes restrições:

$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$

$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$

$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$

$g_4(x) = -8x_1 + x_{10} \leq 0$

$g_5(x) = -8x_2 + x_{11} \leq 0$

$g_6(x) = -8x_3 + x_{12} \leq 0$

$g_7(x) = -2x_4 - x_{15} + x_{10} \leq 0$

$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0$

$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0$

Restrições de domínio:

$0 \leq x_i \leq 1, i = 1, \ldots, 9$

$0 \leq x_i \leq 100, i = 10, 11, 12$

$0 \leq x_i \leq 1, i = 13$

A função G1 é quadrática, de dimensão treze, e há nove restrições lineares (seis restrições ativas no ótimo global). Seu mínimo global é $x^* = (1,1,1,1,1,1,1,1,1,3,3,3,1)$ e $G1(x^*) = -15$.

b) Função G2

Maximizar:

$$G2(x) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

Sujeito às seguintes restrições:

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 0.75n \leq 0$$

Restrições de domínio:

$0 \leq x_i \leq 10, i = 1, \ldots, n; n = 20$

A função G2 é não linear, de dimensão vinte, e o problema apresenta uma restrição linear e uma não linear ativa na origem. Seu máximo global desconhecido, estando em algum lugar perto da origem, e o máximo encontrado é em $f(x^*) = 0.803619$.

Sua representação para o caso de duas variáveis é mostrada na Figura 36, onde foi também desconsiderada a restrição linear (não ativa na região próxima à origem). Algumas potenciais dificuldades de se resolver esse *benchmark* estão ilustradas na figura, onde foram atribuídos valor zero a pontos inválidos. Esse foi o primeiro caso em que a ideia de explorar somente a fronteira entre o espaço de busca válido e inválido foi usada.

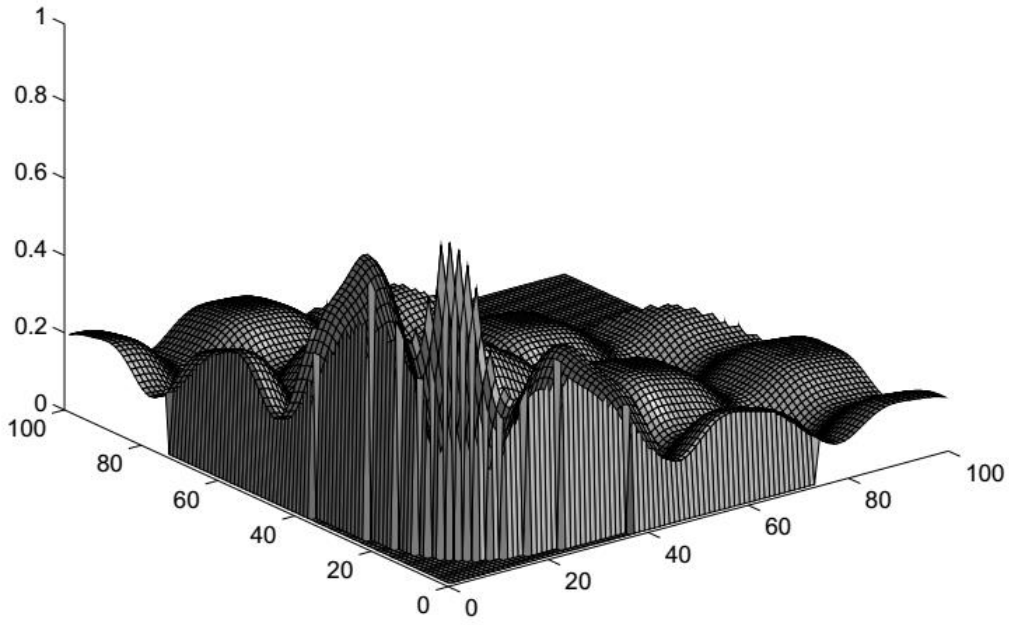


Figura 36. Gráfico da função G2 para *n*=2. Soluções inválidas atribuídas a zero.

c) Função G3

Maximizar:

$$G3(x) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$$

Sujeito às seguintes restrições:

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

Restrições de domínio:

$$0 \leq x_i \leq 1, i = 1, \ldots, n; n = 10$$

A função G3 é polinomial, de dimensão dez e há uma restrição não linear ativa no ótimo. Seu máximo global é $x_i^* = 1/\sqrt{n}$ e $G3(x^*) = 1$.

d) Função G4

Minimizar:

$$G4(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792{,}141$$

Sujeito às seguintes restrições:

$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$

$$g_2(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \leq 0$$

$$g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 - 0.0021813x_3^2 - 110 \leq 0$$

$$g_4(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$

$$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 - 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

Restrições de domínio:

$$78 \leq x_1 \leq 102$$

$$33 \leq x_2 \leq 45$$

$$27 \leq x_i \leq 45, i = 3, 4, 5$$

A função G4 é quadrática, de dimensão cinco, com seis restrições não lineares (duas restrições ativas no ótimo global). Seu mínimo global é $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ e $G4(x^*) = -30665.539$.

e) Função G5

Minimizar:

$$G5(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (\frac{0.000002}{3})x_2^3$$

Sujeito às seguintes restrições:

$$g_1(x) = -x_4 + x_3 - 0.55 \leq 0$$

$$g_2(x) = -x_3 + x_4 - 0.55 \leq 0$$

$$h_3(x) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(x) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(x) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

Restrições de domínio:

$$0 \leq x_i \leq 1200, i = 1, 2$$

$$-0.55 \leq x_i \leq 0.55, i = 3, 4$$

A função G5 é cúbica, tem dimensão quatro, e há duas restrições lineares e três não lineares (três restrições ativas no ótimo global). Seu mínimo global é $x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ e $G5(x^*) = 5126.4981$.

f) Função G6

Minimizar:

$$G6(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

Sujeito às seguintes restrições:

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$

$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 + 82.81 \leq 0$$

Restrições de domínio:

$$13 \leq x_1 \leq 100$$

$$0 \leq x_2 \leq 100$$

A função G6 é cúbica, tem dimensão dois e existem duas restrições não lineares ativas no ótimo global. Seu mínimo global é $x^* = (14.095, 0.84296)$ e $G5(x^*) = -6961.81388$. A Figura 37 ilustra o *benchmark* G6.

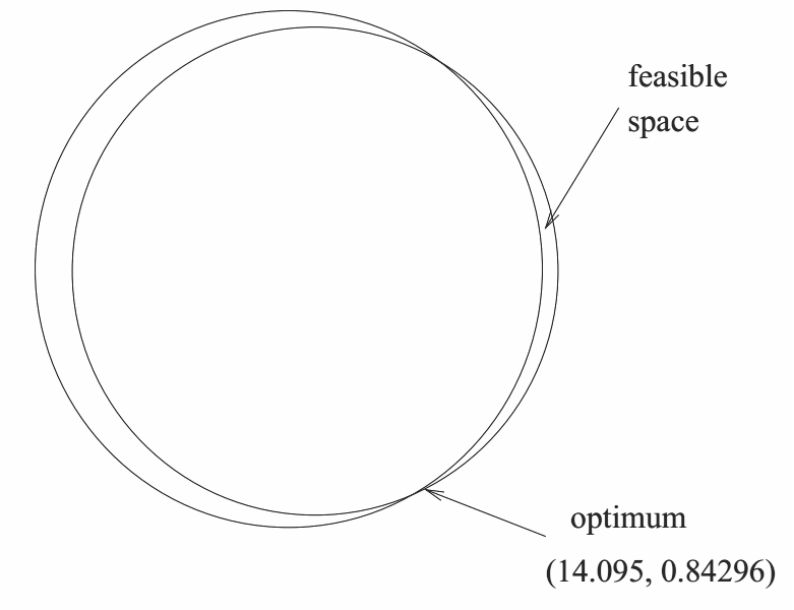


Figura 37. Espaço válido para função G6.

g) Função G7

Minimizar:

$$G7(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

Sujeito às seguintes restrições:

$$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$

$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$

$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$

$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$

$$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$

$$g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$

$$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

Restrições de domínio:

$$-10 \leq x_i \leq 10, i = 1, \dots, 10$$

A função G7 é quadrática, tem dimensão dez, com três restrições lineares e cinco não lineares (seis restrições ativas no ótimo global). Seu mínimo global é $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ e $G7(x^*) = 24.3062091$.

h) Função G8

Maximizar:

$$G8(x) = \frac{sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

Sujeito às seguintes restrições:

$g_1(x) = x_1^2 - x_2 + 1 \leq 0$

$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$

Restrições de domínio:

$0 \leq x_i \leq 10, i = 1{,}2$

A função G8 é não linear, de dimensão dois, e há duas restrições não lineares (nenhuma ativa no ótimo global). Seu máximo global é $x^* = (1.2279713, 4.2453733)$ e $G8(x^*) = 0.095825$. Apresenta muitos ótimos locais; os picos maiores estão ao longo do eixo x, entretanto, na região válida, apresenta dois ótimos locais de valores muito próximo ao máximo global.

i) Função G9

Minimizar:

$$G9(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

Sujeito às seguintes restrições:

$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$

$g_2(x) = -228 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$

$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$

$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$

Restrições de domínio:

$-10 \leq x_i \leq 10, i = 1, \ldots, 7$

A função G9 é polinomial, tem dimensão sete e há quatro restrições não lineares (duas ativas no ótimo global). Seu máximo global é:

$x^* =$ $(2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ e $G9(x^*) = 680.6300573$.

j) Função G10

Minimizar:

$G10(x) = x_1 + x_2 + x_3$

Sujeito às seguintes restrições:

$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$

$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$

$g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$

$g_4(x) = x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0$

$g_5(x) = x_2 x_7 - 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0$

$g_6(x) = x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0$

Restrições de domínio:

$100 \leq x_1 \leq 10000$

$1000 \leq x_i \leq 10000, i = 2, 3$

$10 \leq x_i \leq 1000, i = 4, \ldots, 8$

A função G10 é linear, tem dimensão oito e há três restrições lineares e três não lineares (todas ativas no ótimo global). Seu mínimo global é:

$x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$ e $G10(x^*) = 7049.3307$.

k) Função G11

Minimizar:

$G11(x) = x_1^2 + (x_2 - 1)^2$

Sujeito às seguintes restrições:

$h(x) = x_2 - x_1^2 = 0$

Restrições de domínio:

$-1 \leq x_i \leq 1, i = 1, 2$

A função G11 é quadrática, bidimensional e há uma restrição não linear ativa no ótimo global. Seu mínimo global é $x^* = (\pm 1/\sqrt{2}, 1/2)$ e $G11(x^*) = 0.75$.

l) Função G12

Maximizar:

$G12(x) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$

Sujeito às seguintes restrições:

$g(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$

Restrições de domínio:

$0 \leq x_i \leq 10, i = 1, 2, 3$

$p, q, r = 1, 2, \ldots, 9$

A região válida da função G12 consiste em 9^3 esferas disjuntas. Um ponto (x_1, x_2, x_3) é válido se e somente se existem p, q, r de forma que a inequação definida seja válida. A função é quadrática, e possui 9^3 restrições não lineares, nenhuma ativa no ótimo global. Seu máximo global é $x^* = (5, 5, 5)$ e $G12(x^*) = 1$.

m) Função G13

Minimizar:

$G13(x) = e^{x_1 x_2 x_3 x_4 x_5}$

Sujeito às seguintes restrições:

$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$

$h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0$

$h_3(x) = x_1^3 + x_2^3 + 1 = 0$

Restrições de domínio:

$-2.3 \leq x_i \leq 2.3, i = 1, 2$

$-3.2 \leq x_i \leq 3.2, i = 3, 4, 5$

A função G13 é exponencial e tem três restrições não lineares (todas ativas no ótimo global). Seu mínimo global é:

$x^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$ e $G13(x^*) = 0.0539498$.

n) Função G14

Minimizar:

$$G14(x) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$$

Sujeito às seguintes restrições:

$h_1(x) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$

$h_2(x) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$

$h_3(x) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$

Restrições de domínio:

$0 \leq x_i \leq 10, i = 1, \ldots, 10$

Onde $c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.1$, $c_8 = -10.708$, $c_9 = -26.662$, $c_{10} = -22.179$.

Seu mínimo global é:

$$x^* = (0.0406684113216282, 0.147721240492452, 0.783205732104114, 0.00141433931889084, 0.485293636780388, 0.000693183051556082, 0.0274052040687766, 0.0179509660214818, 0.0373268186859717, 0.0968844604336845)$$

$$G14(x^*) = -47.7648884594915.$$

o) Função G15

Minimizar:

$$G15(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

Sujeito às seguintes restrições:

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

Restrições de domínio:

$$0 \le x_i \le 10, i = 1, 2, 3$$

Seu mínimo global é:

$$x^* = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921)$$

$$G15(x^*) = 961.715022289961$$

p) Função G16

Minimizar:

$$G16(x) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17}$$

Sujeito às seguintes restrições:

$$g_1(x) = \frac{0.28}{0.72}y_5 - y_4 \le 0$$

$$g_2(x) = x_3 - 1.5x_2 \le 0$$

$$g_3(x) = 3496\frac{y_2}{c_{12}} - 21 \le 0$$

$$g_4(x) = 110.6 + y_1 - \frac{62212}{c_{17}} - 21 \leq 0$$

$$g_5(x) = 213.1 - y_1 \leq 0$$

$$g_6(x) = y_1 - 405.23 \leq 0$$

$$g_7(x) = 17.505 - y_2 \leq 0$$

$$g_8(x) = y_2 - 1053.6667 \leq 0$$

$$g_9(x) = 11.275 - y_3 \leq 0$$

$$g_{10}(x) = y_3 - 35.03 \leq 0$$

$$g_{11}(x) = 214.228 - y_4 \leq 0$$

$$g_{12}(x) = y_4 - 665.585 \leq 0$$

$$g_{13}(x) = 7.458 - y_5 \leq 0$$

$$g_{14}(x) = y_5 - 584.463 \leq 0$$

$$g_{15}(x) = 0.961 - y_6 \leq 0$$

$$g_{16}(x) = y_6 - 265.916 \leq 0$$

$$g_{17}(x) = 1.612 - y_7 \leq 0$$

$$g_{18}(x) = y_7 - 7.046 \leq 0$$

$$g_{19}(x) = 0.146 - y_8 \leq 0$$

$$g_{20}(x) = y_8 - 0.222 \leq 0$$

$$g_{21}(x) = 107.99 - y_9 \leq 0$$

$$g_{22}(x) = y_9 - 273.366 \leq 0$$

$$g_{23}(x) = 922.693 - y_{10} \leq 0$$

$$g_{24}(x) = y_{10} - 1286.105 \leq 0$$

$$g_{25}(x) = 926.832 - y_{11} \leq 0$$

$$g_{26}(x) = y_{11} - 1444.046 \leq 0$$

$$g_{27}(x) = 18.766 - y_{12} \leq 0$$

$$g_{28}(x) = y_{12} - 537.141 \leq 0$$

$$g_{29}(x) = 1072.163 - y_{13} \leq 0$$

$$g_{30}(x) = y_{13} - 3247.039 \leq 0$$

$$g_{31}(x) = 8961.448 - y_{14} \leq 0$$

$$g_{32}(x) = y_{14} - 26844.086 \leq 0$$

$$g_{33}(x) = 0.063 - y_{15} \leq 0$$

$$g_{34}(x) = y_{15} - 0.386 \leq 0$$

$$g_{35}(x) = 71084.33 - y_{16} \leq 0$$

$$g_{36}(x) = -140000 + y_{16} \leq 0$$

$$g_{37}(x) = 2802713 - y_{17} \leq 0$$

$$g_{38}(x) = y_{17} - 12146108 \leq 0$$

Restrições de domínio:

$704.4148 \leq x_1 \leq 906.3855$

$68.6 \leq x_2 \leq 288.88$

$0 \leq x_3 \leq 134.75$

$193 \leq x_4 \leq 287.0966$

$25 \leq x_5 \leq 84.1988$

Onde:

$y_1 = x_2 + x_3 + 41.6$

$c_1 = 0.024x_4 - 4.62$

$y_2 = \frac{12.5}{c_1} + 12$

$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$

$c_3 = 0.052x_1 + 78 + 0.002377y_2x_1$

$y_3 = \frac{c_2}{c_3}; y_4 = 19y_3$

$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3$

$c_5 = 100x_2$

$c_6 = x_1 - y_3 - y_4$

$c_7 = 0.950 - \frac{c_4}{c_5}$

$y_5 = c_6c_7$

$y_6 = x_1 - y_5 - y_4 - y_3$

$c_8 = (y_5 + y_4)0.995$

$y_7 = \frac{c_8}{y_1}$

$y_8 = \frac{c_8}{3798}$

$c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153$

$y_9 = \frac{96.82}{c_9} + 0.321y_1$

$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$

$y_{11} = 1.71x_1 - 0.452y_4 + 0.58y_3$

$c_{10} = \frac{12.3}{752.3}$

$c_{11} = (1.75y_2)(0.995x_1)$

$c_{12} = 0.995y_{10} + 1998$

$$y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$$

$$y_{13} = c_{12} - 1.75y_2$$

$$y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5}$$

$$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$$

$$y_{15} = \frac{y_{13}}{c_{13}}$$

$$y_{16} = 148000 - 331000y_{15} + 40y_{13} + 61y_{15}y_{13}$$

$$c_{14} = 2324y_{10} - 28740000y_2$$

$$y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$$

$$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$$

$$c_{16} = 1.104 - 0.72y_{15}$$

$$c_{17} = y_9 + x_5$$

Seu mínimo global é:

$$x^* = \begin{pmatrix} 705.174537070090537, 68.5999999999999943, \\ 102.899999999999991, 282.324931593660324, 37.5841164258054832 \end{pmatrix}$$

$$G16(x^*) = -1.90515525853479$$

q) Função G17

Minimizar:

$$G17(x) = f_1(x_1) + f_2(x_2)$$

$$f_1(x_1) = \begin{cases} 30x_1, & 0 \le x_1 < 300 \\ 31x_1, & 300 \le x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2, & 0 \le x_2 < 100 \\ 29x_2, & 100 \le x_2 < 200 \\ 30x_2, & 200 \le x_2 < 1000 \end{cases}$$

Sujeito às seguintes restrições:

$$h_1(x) = -x_1 + 300 - \frac{x_3 x_4}{131.078}\cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\cos(1.47588) = 0$$

$$h_2(x) = -x_2 - \frac{x_3 x_4}{131.078}\cos(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\cos(1.47588) = 0$$

$$h_3(x) = -x_5 - \frac{x_3 x_4}{131.078}\sin(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078}\sin(1.47588) = 0$$

$$h_4(x) = 200 - \frac{x_3 x_4}{131.078}\sin(1.48477 - x_6) + \frac{0.90798 x_3^2}{131.078}\sin(1.47588) = 0$$

Restrições de domínio:

$0 \leq x_1 \leq 400$

$0 \leq x_2 \leq 1000$

$340 \leq x_3 \leq 420$

$340 \leq x_4 \leq 420$

$-1000 \leq x_5 \leq 1000$

$0 \leq x_6 \leq 0.5236$

Seu mínimo global é:

$$x^* = \begin{pmatrix} 201.784467214523659, 99.9999999999999005, 383.071034852773266, \\ 420, -10.9076584514292652, 0.0731482312084287128 \end{pmatrix}$$

$$G17(x^*) = 8853.53967480648$$

r) Função G18

Maximizar: $f(\vec{x}) = -0.5(x_1 x_4 - x_2 x_3 + x_3 x_9 - x_5 x_9 + x_5 x_8 - x_6 x_7)$

Sujeito às seguintes restrições:

$$g_1(\vec{x}) = 1 - x_3^2 - x_4^2 \geq 0$$

$$g_2(\vec{x}) = 1 - x_9^2 \geq 0$$

$$g_3(\vec{x}) = 1 - x_5^2 - x_6^2 \geq 0$$

$$g_4(\vec{x}) = 1 - x_1^2 - (x_2 - x_9)^2 \geq 0$$

$$g_5(\vec{x}) = 1 - (x_1 - x_5)^2 - (x_2 - x_6)^2 \geq 0$$

$$g_6(\vec{x}) = 1 - (x_1 - x_7)^2 - (x_2 - x_8)^2 \geq 0$$

$$g_7(\vec{x}) = 1 - (x_3 - x_5)^2 - (x_4 - x_6)^2 \geq 0$$

$$g_8(\vec{x}) = 1 - (x_3 - x_7)^2 - (x_4 - x_8)^2 \geq 0$$

$$g_9(\vec{x}) = 1 - (x_7)^2 - (x_8 - x_9)^2 \geq 0$$

$$g_{10}(\vec{x}) = x_1 x_4 - x_2 x_3 \geq 0$$

$$g_{11}(\vec{x}) = x_3 x_9 \geq 0$$

$$g_{12}(\vec{x}) = -x_5 x_9 \geq 0$$

$$g_{13}(\vec{x}) = x_5 x_8 - x_6 x_7 \geq 0$$

Onde os limites são $-1 \leq x_i \leq 1$ $(i = 1, \cdots, 8)$ e $0 \leq x_9 \leq 1$. O ótimo global está em $x^* =$

$(-0.657776192427943163; -0.153418773482438542; 0.323413871675240938;$
$-0.946257611651304398; -0.657776194376798906; -0.753213434632691414;$
$0.323413874123576972; -0.346462947962331735; 0.59979466285217542)$

e $f(x^*) = -0.8660$.

s) Função G19

Maximizar: $f(\vec{x}) = \sum_{i=1}^{10} b_i x_i - \sum_{j=1}^{5} \sum_{i=1}^{5} c_{ij} x_{10+i} x_{10+j} - 2\sum_{j=1}^{5} d_j x_{10+j}^3$

sujeito à:

$$g_j(\vec{x}) = 2\sum_{i=1}^{5} c_{ij} x_{10+i} + 3d_j x_{10+j}^2 e_j - \sum_{i=1}^{10} a_{ij} x_i \geq 0 \; ; \; j = 1, \cdots, 5$$

Onde os limites são $0 \leq x_i \leq 10$ $(i = 1, \cdots, 15)$. O ótimo global está em x* = (1.66991341326291344e - 17; 3.95378229282456509e - 16; 3.94599045143233784; 1.06036597479721211e - 16; 3.2831773458454161; 9.99999999999999822; 1.12829414671605333e-17; 1.2026194599794709e - 17; 2.50706276000769697e- 15; 2.24624122987970677e - 15; 0.370764847417013987; 0.278456024942955571; 0.523838487672241171; 0.388620152510322781; 0.298156764974678579) onde $f(x^*) = -32.386$.

j	1	2	3	4	5
e_j	-15	-27	-36	-18	-12
c_{ij}	30	-20	-10	32	-10
c_{ij}	-20	39	-6	-31	32
c_{3j}	-10	-6	10	-6	-10
c_{4j}	32	-31	-6	39	-20
c_{5j}	-10	32	-10	-20	-30
d_j	4	8	10	6	2
a_{1j}	-16	2	0	1	0
a_{2j}	0	-2	0	0.4	2
a_{3j}	-3.5	0	2	0	0
a_{4j}	0	-2	0	-4	-1
a_{5j}	0	-9	-2	1	-2.8

a_{6j}	2	0	-4	0	0
a_{7j}	-1	-1	-1	-1	-1
a_{8j}	-1	-2	-3	-2	-1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1

b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
-40	-2	-0.25	-4	-4	-1	-40	-60	5	1

t) Função G20

Minimizar: $f(\vec{x}) = \sum_{i=1}^{24} a_i x_i$ sujeito a:

$$h_i(\vec{x}) = \frac{x_{i+12}}{b_{i+12}\sum_{j=13}^{24}\frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12}\frac{x_j}{b_j}} = 0 \quad i = 1, \cdots, 12$$

$$h_{13}(\vec{x}) = \sum_{i=1}^{24} x_i - 1 = 0$$

$$h_{13}(\vec{x}) = \sum_{i=1}^{12} \frac{x_i}{d_i} + 0.7302(530)\left(\frac{14.7}{40}\right)\sum_{i=13}^{24}\frac{x_i}{d_i} - 1.671 = 0$$

$$g_i(\vec{x}) = -\frac{x_i + x_{i+12}}{\sum_{j=1}^{24} x_j + e_i} \geq 0 \quad i = 1, 2, 3$$

$$g_k(\vec{x}) = -\frac{x_{k+3} + x_{k+15}}{\sum_{j=1}^{24} x_j + e_k} \geq 0 \quad k = 4, 5, 6$$

Onde os limites são $0 \leq x_i \leq 1\ (i = 1, \cdots, 24)$. O ótimo global está em $x^* = (9.537E-07, 0, 4.215E-03, 1.039E-04, 0, 0, 2.072E-01, 5.979E-01, 1.298E-01, 3.350e-02, 1.711E-02, 8.427E-03, 4.657E-10, 0, 0, 0, 0, 0, 2.868E-04, 1.193E-03, 8.332E-05, 1.239E-04, 2.070E-05, 1.829E-05)$ e $f(x^*) = 0.09670$.

i	a_i	b_i	c_i	d_i	d_i
1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.20	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.10	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	

10	0.18	82.507	17.7	64.517	
11	0.10	46.07	0.85	49.4	
12	0.09	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.20	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.10	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.10	46.07			
24	0.09	60.097			

u) Função G21

Minimizar: $f(\vec{x}) = (x_1)$ sujeito a:

$$g_1(\vec{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$$
$$h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$
$$h_2(\vec{x}) = 100x_2 + 155.365x_4 - 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$
$$h_3(\vec{x}) = -x_5 + \ln(-x_4 + 900) = 0$$
$$h_4(\vec{x}) = -x_6 + \ln(x_4 + 300) = 0$$
$$h_5(\vec{x}) = -x_7 + \ln(-2x_4 + 700) = 0$$

Onde os limites são $0 \leq x_1 \leq 1000, 0 \leq x_2 \leq 40, 0 \leq x_3 \leq 40, 100 \leq x_4 \leq 300,\ 6.3 \leq x_5 \leq 6.7, 5.9 \leq x_6 \leq 6.4\ e\ 4.5 \leq x_7 \leq 6.25$. O ótimo global está em $x^* =$ (193.724510070034967; 5.56944131553368433e - 27; 17.3191887294084914; 100.047897801386839; 6.68445185362377892; 5.99168428444264833; 6.21451648886070451) e $f(x^*) = 193.7783493$.

v) Função G22

Minimizar: $f(\vec{x}) = (x_1)$ sujeito a :

$$g_1(\vec{x}) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0$$
$$h_1(\vec{x}) = x_5 - 10000x_8 + 1E07 = 0$$

$$h_2(\vec{x}) = x_6 - 10000x_8 + 10000x_9 = 0$$
$$h_3(\vec{x}) = x_7 - 10000x_9 - 5E07 = 0$$
$$h_4(\vec{x}) = x_5 - 10000x_{10} - 4.4E07 = 0$$
$$h_5(\vec{x}) = x_6 - 10000x_{11} - 6.6E07 = 0$$
$$h_6(\vec{x}) = x_7 - 10000x_{12} - 6.6E07 = 0$$
$$h_7(\vec{x}) = x_5 - 120x_2x_{13} = 0$$
$$h_8(\vec{x}) = x_6 - 80x_3x_{14} = 0$$
$$h_9(\vec{x}) = x_7 - 40x_4x_{15} = 0$$
$$h_{10}(\vec{x}) = x_8 - x_{11} + x_{16} = 0$$
$$h_{11}(\vec{x}) = x_9 - x_{12} + x_{17} = 0$$
$$h_{12}(\vec{x}) = -x_{18} + \ln(x_{10} - 100) = 0$$
$$h_{13}(\vec{x}) = -x_{19} + \ln(-x_8 + 300) = 0$$
$$h_{14}(\vec{x}) = -x_{20} + \ln(x_{16}) = 0$$
$$h_{15}(\vec{x}) = -x_{21} + \ln(-x_9 + 400) = 0$$
$$h_{16}(\vec{x}) = -x_{22} + \ln(x_{17}) = 0$$
$$h_{17}(\vec{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$$
$$h_{18}(\vec{x}) = -x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$$
$$h_{19}(\vec{x}) = -x_9 - x_{12} - 4.60517\, x_{15} + x_{15}x_{22} + 100 = 0$$

Onde os limites são $0 \leq x_1 \leq 2000, 0 \leq x_i \leq 1E06\ I = (2,3,4), 0 \leq x_i \leq 4E07\ i = (5,6,7), 100 \leq x_8 \leq 299.99, 100 \leq x_9 \leq 399.99, 100.01 \leq x_{10} \leq 300, 100 \leq x_{11} \leq 400, 100 \leq x_{12} \leq 600, 0 \leq x_{10} \leq 500\ i(13,14,15), 0.01 \leq x_{16} \leq 300, 0.01 \leq x_{17} \leq 400\ e - 4.7 \leq x_i \leq 6.25\ i = (18, \cdots, 22)$. O ótimo global está em x^* =(236.430975504001054; 135.82847151732463; 204.818152544824585; 6446.54654059436416; 3007540.83940215595; 4074188.65771341929; 32918270.5028952882; 130.075408394314167; 170.817294970528621; 299.924591605478554; 399.258113423595205; 330.817294971142758; 184.51831230897065; 248.64670239647424; 127.658546694545862; 269.182627528746707; 160.000016724090955; 5.29788288102680571; 5.13529735903945728; 5.59531526444068827; 5.43444479314453499; 5.07517453535834395), onde $f(x^*) = 12812.5$.

w) Função G23

Minimizar: $f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 - x_7)$ sujeito a:

$$h_1(\vec{x}) = x_1 + x_2 - x_3 - x_4 = 0$$
$$h_2(\vec{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$$

$$h_3(\vec{x}) = x_3 + x_6 - x_5 = 0$$
$$h_4(\vec{x}) = x_4 + x_7 - x_8 = 0$$
$$g_1(\vec{x}) = x_9 x_3 + 0.02x_6 - 0.025x_5 \leq 0$$
$$g_2(\vec{x}) = x_9 x_4 + 0.02x_7 - 0.015x_8 \leq 0$$

Onde os limites são $0 \leq x_i \leq 300$ $i = (1, 2, 6), 0 \leq x_i \leq 100$ $i = (3, 5, 7), 0 \leq x_i \leq 200$ $i = (4, 8)$ e $0.01 \leq x_9 \leq 0.03$. A melhor solução conhecida tem o valor função objeto de $f(x^*) = -400.0551$.

x) Função G24

Minimizar: $f(\vec{x}) = (-x_1 - x_2)$ sujeito a:

$$g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$
$$g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

Onde os limites são $0 \leq x_1 \leq 3, 0 \leq x_2 \leq 4$. O ótimo está em $x^* = (2.3295;\ 3.17846)$ onde $f(x^*) = -5.50796$.

Apêndice B – Funções de Avaliação para o Estudo de Caso Real

Função de Avaliação

A função de avaliação é responsável por fornecer o valor da avaliação de cada solução proposta pelo algoritmo. O valor de avaliação de uma solução, ou seja, de uma alternativa de poços, é dado pelo seu VPL, que é calculado a partir do perfil de produção, que por sua vez é gerado por um simulador de reservatório e por parâmetros econômicos informados ao modelo. O cálculo do VPL de um estudo é descrito em detalhes abaixo.

VPL de um Estudo

$$VPL_{Estudo} = \sum_{i=1}^{N} VPL_{Plat_i}$$

Onde:

VPL_{Plat}: valor presente líquido da plataforma,

N: número de plataformas.

VPL da Plataforma:

$$VPL_{Plat} = \sum_{j=1}^{M} VP_{Poço_j} - \sum_{j=1}^{M} C_{Poço_j} - C_{Plat}$$

Onde:

M: número de poços na plataforma,

$VP_{Poço}$: valor presente do poço,

$C_{Poço}$: custo de desenvolvimento do poço,

C_{Plat}: custo da plataforma.

Custo da plataforma:

$$C_{Plat} = CF + LDA^{máx} \times C_{LDA} + Q_O \times C_O + Q_{AP} \times C_{AP} + Q_L \times C_L + Q_{AI} \times C_{AI} + Q_{GP} \times C_{GP} + Q_{GI} \times C_{GI} + Q_{SLOT} \times C_{SLOT}$$

Onde:

CF: custo fixo da plataforma,

$LDA^{máx}$: limite de lâmina d'água,

C_{LDA}: custo em função da lâmina d'água,

Q_O: capacidade máxima de produção de óleo,

C_O: custo de produção de óleo,

Q_{AP}: capacidade máxima de produção de água,

C_{AP}: custo de produção de água,

Q_L: capacidade máxima de produção de líquido,

C_L: custo de produção de líquido,

Q_{AI}: capacidade máxima de injeção de água,

C_{AI}: custo de injeção de água,

Q_{GP}: capacidade máxima de produção de gás,

C_{GP}: custo de produção de gás,

Q_{GI}: capacidade máxima de injeção de gás,

C_{GI}: custo de injeção de gás,

Q_{SLOT}: capacidade máxima de slot,

C_{SLOT}: custo por número de slot.

Custo de Desenvolvimento do Poço:

$$C_{Poço} = CP_{Poço} + CL_{Poço} + CA_{Poço}$$

Onde:

$CP_{Poço}$: custo de perfuração do poço,

$CL_{Poço}$: custo da linha do poço,

$CA_{Poço}$: custo de abandono do poço.

Custo de Perfuração do Poço:

$$CP_{Poço} = \sum_{k=1}^{3}\sum_{l=1}^{3}(CP_k^{nc} \times L_k^{nc} + CP_{kl}^{c} \times L_{kl}^{c})$$

Onde:

k: tipo do poço, $k \in \{1,2,3\}$,

$k = 1$: representa um poço produtor,

$k = 2$: representa um poço injetor de água,

$k = 3$: representa um poço injetor (de gás, cíclico ou solvente),

l: intervalo angular, $l \in \{1,2,3\}$,

$l = 1$: intervalo angular entre 0º e ângulo inicial,

$l = 2$: intervalo angular entre ângulo inicial e ângulo intermediário,

$l = 3$: intervalo angular entre ângulo intermediário e 90º,

CP_k^{nc}: custo de perfuração do trecho não canhoneado por metro de poço do tipo k,

L_k^{nc}: comprimento do trecho não canhoneado do poço do tipo k,

CP_{kl}^{c}: custo de perfuração do trecho canhoneado por metro de poço do tipo k com inclinação no intervalo l,

L_{kl}^{c}: comprimento do trecho canhoneado do poço do tipo k com inclinação no intervalo l.

Custo de Abandono do Poço:

$$CA_{Poço} = (CP_{Poço} \times CA_{\%})$$

Onde:

$CP_{Poço}$: custo de perfuração do poço,

$CA_{\%}$: percentual do custo de perfuração que corresponde ao custo de abandono do poço.

Custo da Linha do Poço:

$$CL_{Poço} = \sum_{k=1}^{3}[(CR \times LA) + (CF_k \times LF_k)]$$

Onde:

CR: custo do *riser*,

LA: lâmina d'água,

CF_k: custo da *flowline* do poço do tipo k.

LF_k: comprimento da *flowline* do poço do tipo k.

$$LF_k = \sqrt{(cw_{xk} - cp_{xm})^2 + \left(cw_{yk} - cp_{ym}\right)^2}$$

Onde:

LF_k: comprimento da *flowline* do poço do tipo k,

cw_{xk} : coordenada x da cabeça do poço do tipo k,

cw_{yk} : coordenada y da cabeça do poço do tipo k,

cp_{xm} : coordenada x da plataforma do tipo m,

cp_{ym} : coordenada y da plataforma do tipo m.

Valor Presente do Poço:

$$VP_{Poço} = \left(VP_{r_{Poço}} - VP_{cop_{Poço}}\right) \times (1 - I)$$

Onde:

$VP_{r_{Poço}}$: valor presente da receita do poço,

$VP_{cop_{Poço}}$: valor presente do custo operacional do poço,

I : alíquota de impostos.

Valor Presente da Receita do Poço:

$$\boldsymbol{VP_{r_{poço}} = \sum_{t_0}^{T} \frac{R^t}{(1 + tma)^{\left(\frac{d^t}{365}\right)}}}$$

Onde:

R^t: receita no tempo t,

t_0: início da produção,

T: tempo total da produção.

tma: taxa mínima de atratividade,

d^t: dia no tempo t.

$$\boldsymbol{R^t = (q_O^t \times p_O^t + q_G^t \times p_G^t) \times (d^t - d^{t-1})}$$

Onde:

q_O^t: vazão da produção de óleo no tempo t,

p_O^t: preço da venda de óleo no tempo t,

q_G^t: vazão da produção de gás no tempo t,

p_G^t: preço da venda de gás no tempo t.

Valor Presente do Custo Operacional do Poço:

$$\boldsymbol{VP_{cop_{poço}}} = \sum_{\boldsymbol{t=1}}^{\boldsymbol{T}} \frac{\boldsymbol{Cop^t}}{(\mathbf{1}+\boldsymbol{tma})^{\left(\frac{\boldsymbol{d^t}}{\mathbf{365}}\right)}}$$

Onde:

Cop^t : custo operacional no tempo t,

tma: taxa mínima de atratividade,

d^t: dia no tempo t.

T: tempo total da produção,

$$\begin{aligned} Cop^t = & \left[C_M \times \left(\frac{d^t - d^{t-1}}{365}\right)\right] + \left[CF_{Prod} \times \left(\frac{d^t - d^{t-1}}{365}\right)\right] \\ & + \left[CF_{Inj} \times \left(\frac{d^t - d^{t-1}}{365}\right)\right] \quad + Ry \times R^t \\ & + (C_O \times q_O^t + C_{GP} \times q_{GP}^t + C_{AP} \times q_{AP}^t + C_{AI} \times q_{AI}^t + C_{GI} \times q_{GI}^t) \\ & \times (d^t - d^{t-1}) \end{aligned}$$

Onde:

C_M: custo de manutenção por poço (anual),

CF_{Prod}: custos fixos de produção por poço (anual),

CF_{Inj}: custos fixos de injeção por poço (anual),

Ry: percentual dos royalties,

R^t: receita no tempo t,

C_O: custos de produção de óleo,

q_O^t: vazão de produção de óleo no tempo t,

C_G: custos de produção de gás,

q_G^t: vazão de produção de gás no tempo t,

C_{AP}: custos de produção de água,

q_{AP}^t: vazão de produção de água no tempo t,

C_{AI}: custos de injeção de água,

q_{AI}^{t}: vazão de injeção de água no tempo t,

C_{GI}: custos de injeção de gás,

q_{GI}^{t}: vazão de injeção de gás no tempo t,

$d(t)$: dia no tempo t.