



ALGORITMOS EVOLUTIVOS APLICADOS A PROBLEMAS ENVOLVENDO
FUNÇÕES COMPUTACIONALMENTE CUSTOSAS EM DOMÍNIOS
RESTRITOS

Rafael de Paula Garcia

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Civil.

Orientadores: Beatriz de Souza Leite Pires de
Lima
Afonso Celso de Castro
Lemonge

Rio de Janeiro
Março de 2018

ALGORITMOS EVOLUTIVOS APLICADOS A PROBLEMAS ENVOLVENDO
FUNÇÕES COMPUTACIONALMENTE CUSTOSAS EM DOMÍNIOS
RESTRITOS

Rafael de Paula Garcia

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ
COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE)
DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR
EM CIÊNCIAS EM ENGENHARIA CIVIL.

Examinada por:

Prof. Beatriz de Souza Leite Pires de Lima, D.Sc.

Prof. Afonso Celso de Castro Lemonge, D.Sc.

Prof. Nelson Francisco Favilla Ebecken, D.Sc.

Prof. Alexandre Gonçalves Evsukoff, D.Sc.

Prof. José Gabriel Rodriguez Carneiro Gomes, D.Sc.

Prof. Marley Maria Bernardes Rebuzzi Vellasco, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2018

Garcia, Rafael de Paula

Algoritmos evolutivos aplicados a problemas envolvendo funções computacionalmente custosas em domínios restritos/Rafael de Paula Garcia. – Rio de Janeiro: UFRJ/COPPE, 2018.

X, 76 p. 29, 7cm.

Orientadores: Beatriz de Souza Leite Pires de Lima
Afonso Celso de Castro Lemonge

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Civil, 2018.

Referências Bibliográficas: p. 40 – 46.

1. Otimização. 2. Algoritmos Evolutivos. 3. Tratamento de restrições. 4. Método de aproximação de funções. I. Lima, Beatriz de Souza Leite Pires de *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

A maior das descobertas científicas foi a descoberta da ignorância. Uma vez que os humanos se deram conta de quão pouco eles sabiam sobre o mundo, eles tiveram um motivo muito bom para ir em busca de conhecimento
(Yuval Noah Harari)

Agradecimentos

À minha família, que foram e são os principais apoiadores desta jornada. É para vocês cada conquista minha.

À Bia, pela orientação, apoio e constante motivação. Esses quatro anos de doutorado foram significativamente facilitados tendo você por perto para mostrar a melhor direção a seguir. Obrigado pela sua generosidade, paciência e amizade.

Ao Afonso Lemonge, que acompanha meus passos acadêmicos desde o mestrado e que inspirou o tema desta tese. Ao professor Breno Jacob, pelas oportunidades e pela parceria.

Às grandes amizades formadas durante o doutorado e às de longa data.

Agradeço à Universidade Federal do Rio de Janeiro, à COPPE e ao Programa de Engenharia Civil, pelo suporte acadêmico.

Ao CNPq e à FAPERJ, pelo apoio financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ALGORITMOS EVOLUTIVOS APLICADOS A PROBLEMAS ENVOLVENDO
FUNÇÕES COMPUTACIONALMENTE CUSTOSAS EM DOMÍNIOS
RESTRITOS

Rafael de Paula Garcia

Março/2018

Orientadores: Beatriz de Souza Leite Pires de Lima
Afonso Celso de Castro Lemonge

Programa: Engenharia Civil

A aplicação de algoritmos evolutivos na otimização de problemas reais e complexos de engenharia tem se mostrado bastante eficiente. Porém, como a maioria destes problemas são definidos por funções objetivo e restrições caras computacionalmente, novas técnicas de modelagem e tratamento de restrições têm sido propostas. Neste cenário, esta tese propõe um tratamento de restrições denominado *Multiple Constraint Ranking* (MCR) e um algoritmo de aproximação de funções baseado em similaridade. Eles auxiliam algoritmos evolutivos na busca de soluções ótimas em problemas de otimização com restrições e função objetivo dispendiosa computacionalmente. O MCR calcula a aptidão das soluções segundo a soma de suas posições em várias filas, com base nos valores da função objetivo, da violação em cada restrição e do número de restrições violadas. A aproximação baseada em similaridade estima o valor da função objetivo de uma solução por uma média ponderada dos valores originais da função objetivo de soluções “vizinhas” pelas distâncias. Tais soluções são selecionadas a partir de um banco de dados, cuja atualização é baseada na contribuição das soluções no processo de aproximação. Três algoritmos são propostos: i) MCR acoplado a um algoritmo genético; ii) evolução diferencial utilizando aproximação da função objetivo por similaridade; e, iii) evolução diferencial assistida pelo MCR e a aproximação de funções. Eles foram submetidos a problemas complexos sugeridos pelas competições do IEEE-CEC e em problemas clássicos de engenharia estrutural. Seus resultados foram comparados com relevantes algoritmos da literatura, onde comprovou-se a robustez de todos eles.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

EVOLUTIONARY ALGORITHMS APPLIED TO PROBLEMS INVOLVING
COMPUTATIONALLY EXPENSIVE FUNCTIONS IN CONSTRAINED
DOMAINS

Rafael de Paula Garcia

March/2018

Advisors: Beatriz de Souza Leite Pires de Lima
Afonso Celso de Castro Lemonge

Department: Civil Engineering

The use of evolutionary algorithms in the optimization of real and complex engineering problems has proved to be quite efficient. However, since most of these problems are defined by expensive objective function and constraints, new modeling and constraint handling techniques have been proposed. In this scenario, this thesis proposes a constraint handling technique called Multiple Constraint Ranking (MCR) and a similarity-based surrogate algorithm. They assist evolutionary algorithms in the search for optimal solutions in optimization problems in which both objective function and constraints are costly. The MCR calculates the fitness of the solutions according to the sum of their positions in several queues, based on the values of the objective function, the violation in each constraint and the number of constraints violated. The similarity-based approach estimates the value of the objective function of a solution by a weighted average of the original objective function values of “neighboring” solutions by their distances. Such solutions are selected from a database, whose updating is based on the contribution of the solutions in the approximation process. Three algorithms are proposed: i) MCR coupled with a genetic algorithm; ii) differential evolution using similarity-based approximation; and, iii) differential evolution assisted by the MCR and the approximation. They were applied to complex problems suggested by the IEEE-CEC competitions and classical structural engineering problems. Their results were compared with relevant algorithms in the literature, where were proved the robustness of all of them.

Sumário

Lista de Tabelas	x
1 Introdução	1
1.1 Motivação e objetivos	3
1.2 Contribuições	4
1.3 Organização da tese	5
2 Problema de otimização	6
2.1 Algoritmos evolutivos em problemas de otimização com restrições . . .	8
2.2 Evolução assistida por modelos substitutos	11
3 Um tratamento de restrições baseado em múltiplos rankings	13
3.1 Experimentos computacionais com o MCR	15
3.2 Conclusões: MCR	16
4 Um modelo de aproximação baseado em similaridade com gerenciamento do banco de dados via mérito	18
4.1 Experimentos computacionais: SADE-kNN	20
4.2 Conclusões: KNN com gerenciamento por mérito	21
5 O acoplamento de uma evolução diferencial, o MCR e o metamodelo baseado em kNN com gerenciamento via mérito	22
5.1 Experimentos computacionais	25
5.1.1 SADE- k NN-MCR acoplado a uma estratégia que evita a estagnação em ótimos locais	27
5.1.2 Estudo comparativo dos resultados	32
5.2 Conclusões: Evolução diferencial assistida pelo k -NN via mérito e MCR	36
6 Conclusões gerais e trabalhos futuros	37
Referências Bibliográficas	40

A	A Surrogate Assisted Differential Evolution to Solve Constrained Optimization Problems	47
B	A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms	54

Lista de Tabelas

3.1	MCR, exemplo ilustrativo: definição do problema	14
3.2	MCR, exemplo ilustrativo: resultados	15
3.3	Problemas de otimização: experimentos computacionais	16
4.1	k NN com gerenciamento do banco de dados por mérito, exemplo ilustrativo: banco de dados definido por 4 soluções (BD_i) avaliadas pela função original (f) para aproximar o valor da função objetivo (f_{approx}) de 2 soluções (S_j) da população corrente, utilizando o sistema de pontuação por mérito para sua atualização	19
5.1	Resultados preliminares encontrados pelo Algoritmo 3 nos problemas G1 e G11	26
5.2	Resultados encontrados pelo SADE- k NN-MCR para o conjunto de funções teste G-Suíte	28
5.3	Resultados comparativos entre o SADE- k NN-MCR e os algoritmos dos capítulos anteriores no conjunto de funções G Suite	35
5.4	Comparativo entre as melhores soluções nos problemas cujas restrições apresentam diferenças de magnitudes	35
5.5	Média do número de avaliações necessárias para alcançar as soluções ótimas em problemas com restrições com diferentes ordens de magnitude	35

Capítulo 1

Introdução

O processo evolutivo da adaptação das espécies ao ambiente natural, a busca coordenada de alimentos por bando de pássaros ou colônia de formigas e a resposta do sistema imunológico à invasão de agentes patógenos são exemplos de sistemas naturais que procuram as melhores opções em ambientes com elevadas complexidades. Baseado no êxito desses mecanismos, vários pesquisadores acreditaram que essas estruturas pudessem servir de inspiração para ambientes computacionais ao tratar problemas de otimização complexos. A partir daí vários algoritmos computacionais foram propostos durante os últimos anos, como o algoritmo genético [1], estratégias evolutivas [2], programação genética [3], colônia de formigas [4], evolução diferencial [5], entre outros.

A aplicação desses algoritmos a problemas reais de otimização se concretizou nas últimas décadas, como na engenharia, na economia, na medicina, entre outros. Os bons resultados alcançados encorajaram vários especialistas a aprimorarem tais algoritmos para que problemas ainda mais complexos pudessem ser atendidos por eles.

Restrições e funções custosas são algumas dificuldades encontradas em diversos problemas reais de otimização. Mesmo na presença desses fatores, que atribuem complexidade à busca pela solução ótima, algoritmos evolutivos estão sendo aplicados mesmo que acoplados a novas metodologias que tratam cada uma dessas complexidades separadamente.

Embora os algoritmos evolutivos tenham sido originalmente desenvolvidos para serem aplicados a problemas sem restrições, eles frequentemente são complementados por técnicas de tratamento de restrições para também resolverem problemas com restrições. Uma vertente muito utilizada dessas técnicas é aquela baseada em penalização. Nela, o problema com restrição é transformado em um problema sem restrição através da definição de uma função de aptidão (ou função *fitness*) que incorpora as violações à função objetivo através de uma função de penalização. BARBOSA e LEMONGE [6] definem, adaptativamente, uma função de penalização

que leva em conta o nível de violação nas restrições e algumas informações coletadas da população. Sua função de aptidão é definida, portanto, por uma soma entre a função de penalização e a função objetivo. Outra importante linha consiste no tratamento de restrições baseado em ranking. Nesse contexto, RUNARSSON e YAO [7] propõem uma técnica que balanceia as funções objetivo e de penalidade por um parâmetro P_f através de um ranking que compara soluções adjacentes. Os mesmos autores, mais tarde, propuseram uma outra técnica que também balanceia as funções objetivo e de penalidade e define uma função aptidão dependente de dois rankings: um de acordo com a função objetivo e, outro, de acordo com o somatório das violações [8]. HO e SHIMIZU [9] propõem um esquema baseado em ranking em que as soluções são classificadas de acordo com as suas posições em três rankings distintos: função objetivo, somatório das violações nas restrições e número de restrições violadas. Recentemente um método baseado em ranking [10] foi proposto organizando soluções em uma fila construída a partir de uma mesclagem de dois rankings, um para soluções factíveis e outro para soluções infactíveis. Uma terceira abordagem para restrições em algoritmos evolutivos está no processo de seleção de soluções para a reprodução. Nessa etapa, uma abordagem que geralmente produz bons resultados prioriza soluções factíveis a infactíveis [11].

Avaliar uma solução candidata no processo de otimização de problemas reais de engenharia é uma tarefa que pode consumir muito recurso e/ou tempo computacional. Utilizar algoritmos evolutivos neste tipo de problema pode ser impraticável, já que eles exigem uma grande quantidade de avaliações. Uma alternativa para lidar com esse impasse é a construção de um modelo de aproximação que seja menos custoso computacionalmente (e eles são chamados de *surrogate* ou *metamodel*) para parcialmente substituírem tais funções que são computacionalmente custosas. Tais modelos de aproximação podem compreender polinômios, função de base radial, rede neural e também aqueles baseados em similaridade.

Em [12] a função objetivo foi aproximada usando um modelo polinomial acoplado a um algoritmo evolutivo. Modelos de aproximação baseados em redes neurais foram utilizados em [13, 14] para reduzir o tempo de processamento utilizado pelo método de elementos finitos dinâmicos para a análise de estruturas *offshore*. Também em [15] uma rede neural multi-camada foi implementada para aproximar a função de aptidão. Outros modelos de aproximação são aplicados na inicialização das soluções da população, nos operadores evolutivos como mutação e *crossover* [16, 17] ou mesmo para aproximar o somatório das violações pelas soluções [18]. Recentemente muitos trabalhos estão considerando o uso de modelos de aproximações baseados em similaridade devido à sua fácil implementação e aos bons resultados alcançados [18–20].

Embora já existam muitos estudos propondo técnicas de tratamento de restrições e modelos de aproximação, ainda há espaço para que novas abordagens sejam pro-

postas e para que mais análises dos métodos já existentes sejam feitas para entender suas reais potencialidades. Sem dúvida, todos os tratamentos de restrições mencionados anteriormente apresentam muitas vantagens em relação a tratamentos padrões [21], como a penalização estática, quando aplicados a problemas complexos de engenharia. No entanto, ainda existem questões e desafios relacionados à definição de boas estratégias para aproximar funções, assim como em técnicas para tratar soluções. Atualmente, a principal questão está em como combinar, em uma função objetivo, os valores relacionados à função objetivo e às violações nas restrições quando, por exemplo, elas possuírem diferentes ordens de magnitudes e/ou unidades. Isso não seria um problema para os casos mais comuns em que os intervalos de valores mínimos e máximos de violação são conhecidos a priori. Esses casos podem ser facilmente tratados pelos procedimentos de normalização. No entanto, existem muitas aplicações para as quais tais informações não estão disponíveis e não podem ser estimadas antecipadamente, incluindo, por exemplo, aplicações práticas relacionadas a estruturas *offshore*, como a aplicação de *risers* conectados a plataformas flutuantes para produção de petróleo *offshore* [22–24], ou a otimização de rotas de gasodutos submarinos [25, 26].

Já no que diz respeito ao uso de modelos de aproximação, o principal desafio atualmente, principalmente ao se utilizar aqueles baseados em similaridade, recai sobre a definição do conjunto de dados (geralmente definido por soluções avaliadas pelo modelo exato) que será utilizado para aproximar as soluções da população corrente. Como as avaliações das soluções serão aproximadas por similaridade, é razoável imaginar que esse banco de dados deve representar bem qualquer solução corrente durante toda a evolução. Porém, em um algoritmo evolutivo a população está em constante renovação, e um banco de dados utilizado em uma geração pode não ter a mesma eficiência para aproximar soluções de uma outra geração. Ou seja, idealmente ele não deve ser estático. Ele deve se ajustar à evolução de acordo com o decorrer das gerações.

1.1 Motivação e objetivos

Alguns problemas reais em engenharia como, por exemplo, a otimização de sistemas estruturais para produção de petróleo *offshore* são bastante complexos. Problemas deste tipo exigem muito recurso computacional para suas simulações pois demandam a utilização de métodos numéricos tais como o método dos elementos finitos e podem incluir análises dinâmicas. Portanto, avaliar uma solução candidata pode ser uma tarefa computacionalmente custosa já que a função objetivo do problema requer o uso dessas simulações. Esse panorama piora ao se utilizar algoritmos populacionais evolutivos já que, a cada iteração, é necessário avaliar todas as soluções candidatas

da população. Além disso, esses problemas estão sujeitos a restrições, que dificultam a busca por soluções ótimas factíveis e aumentam o tempo para a convergência do processo.

Portanto, o objetivo geral desta tese é o de estudar o comportamento de algoritmos evolutivos aplicados a problemas de otimização complexos, que envolvam restrições e funções computacionalmente custosas.

Pretende-se desenvolver e estudar um tratamento de restrições que lide bem com grandes diferenças de magnitudes e/ou unidades das restrições, sabendo-se que este tipo de situação é frequente nos problemas de otimização com restrições. Sobre problemas que envolvam funções custosas, deseja-se aplicar e estudar um modelo substituto baseado em similaridade.

1.2 Contribuições

Esta tese apresenta um algoritmo preciso e robusto para resolver problemas de otimização com restrições, para os quais a avaliação da função objetivo seja cara computacionalmente. A importância deste algoritmo deve-se ao fato de que muitos problemas reais de engenharia são modelados com tais características: domínio restrito e função objetivo dispendiosa.

Este algoritmo é composto, basicamente, por um algoritmo evolutivo, um tratamento de restrições e um metamodelo. O tratamento de restrições proposto nesta tese, capaz de lidar com restrições de igualdade e desigualdade, habilita a aplicação de qualquer algoritmo evolutivo em problemas de otimização cujo domínio de viabilidade seja restrito. Ele atribui o mesmo grau de importância a qualquer restrição do problema, mesmo quando elas possuem diferenças significativas de magnitudes, fato recorrente em problemas de engenharia estrutural, de petróleo, entre outras. Já o metamodelo proposto por esta tese é baseado em modelos de aproximação por similaridade. Estes modelos usualmente definem um banco de dados para aproximar os valores de suas soluções. Nesta tese, este gerenciamento é feito por um esquema baseado em mérito, onde as soluções que mais contribuem para o processo de aproximação tendem a permanecer por mais gerações no banco de dados. Esta abordagem faz com que as soluções sejam aproximadas com mais precisão, aumentando a velocidade de convergência do algoritmo.

Como muitos algoritmos evolutivos, principalmente aqueles inspirados na Evolução Diferencial, possuem problemas de convergência prematura, esta tese considera uma estratégia que auxilia o algoritmo na desvinculação do ótimo local quando ele se estagna por muitas gerações.

1.3 Organização da tese

O Capítulo 2 define um problema de otimização com restrições e custoso, assim como faz um breve estudo do estado da arte envolvendo algoritmos evolutivos para resolver este tipo de problema.

O Capítulo 3 apresenta o MCR (*Multiple Constraint Ranking*), um tratamento de restrições baseado em ranking que foi desenvolvido para lidar, principalmente, com problemas de otimização cujo espaço viável de soluções é definido por funções que, quando avaliadas, retornam valores com diferentes magnitudes. Este capítulo também descreve os experimentos computacionais feitos e publicados com tal tratamento em problemas *benchmark* e de engenharia.

O Capítulo 4 descreve um modelo de aproximação de funções baseado em similaridade cujo banco de dados é gerenciado através de um esquema de mérito. Também são descritos os resultados computacionais realizados, utilizando tal modelo acoplado a uma evolução diferencial.

O acoplamento entre evolução diferencial, MCR e o modelo de aproximação baseado em similaridade foi desenvolvido no Capítulo 5. São apresentados os pseudocódigos e os resultados obtidos nos experimentos computacionais.

As conclusões gerais obtidas nesta tese são apresentadas no Capítulo 6.

A omissão de alguns resultados no corpo desta tese justifica-se pelo fato deles já terem sido publicados em revista e conferências científicas com relevantes impactos internacionais. Isto, porém, não penalizará as análises dos leitores, visto que os artigos serão referenciados e anexados à esta tese.

Capítulo 2

Problema de otimização

Naturalmente, sempre se deseja obter a melhor opção entre todas aquelas disponíveis quando deve-se tomar alguma decisão. Seja ao escolher o trajeto mais curto entre duas cidades, seja reduzindo ao máximo a quantidade de madeira utilizada ao se confeccionar a tampa redonda de uma mesa. Supondo que ambos os objetivos mencionados acima (trajeto mais curto e quantidade mínima de madeira) pudessem ser descritos matematicamente por uma função objetivo $f(\mathbf{x})$, então poderíamos escrever:

$$\text{Min/Max} \quad f(\mathbf{x}) \quad (2.1)$$

$$\mathbf{x} \in D \quad (2.2)$$

onde \mathbf{x} é chamado de variável de decisão e descreve fatores importantes do problema que deve ser otimizado. D representa o domínio definido para as variáveis \mathbf{x} . Por exemplo, $\mathbf{x} = x_1, x_2, \dots, x_r$ pode ser definido por um vetor de variáveis inteiras 0-1 ($D = \{0, 1\}$) que representam cidades intermediárias entre as duas cidades do primeiro exemplo. Ao assumir valor 1 (suponha $x_i = 1$), a rota obrigatoriamente deve passar pela cidade i . Otimizar este problema significa atribuir convenientemente valores 1 àquelas variáveis tais que, passando pelas cidades que elas representam, minimiza-se o caminho entre as duas cidades finais. Já no segundo exemplo, x pode representar, em centímetros, o raio da circunferência da mesa redonda. Nesse caso, x deve assumir o menor valor definido em D .

Dizemos que o problema de otimização definido pelas Equações (2.1) e (2.2) é sem restrições, ou seja, as variáveis do problema podem assumir quaisquer valores definidos no domínio D . Porém, se no primeiro exemplo, não existisse estrada entre as cidades x_2 e x_4 e nem entre as cidades x_3 e x_6 (aqui supondo que $r > 6$), a quantidade de combinações possíveis considerando todas as cidades em D seria diretamente afetada. Em outras palavras, dizemos que foram impostas restrições às

combinações que poderiam ser feitas.

Quando restrições são impostas às variáveis do problema diz-se que se trata de um problema com restrições. Um problema geral de otimização com restrições pode ser formalmente definido considerando um espaço de busca r -dimensional composto por um vetor de variáveis de decisão $\mathbf{x} = (x_1, x_2, \dots, x_r)$ cujas componentes x_i são limitadas inferior e superiormente $[l_k, u_k]$. O objetivo é otimizar uma função objetivo $f(\mathbf{x})$ considerando restrições definidas por desigualdades e igualdades (respectivamente $g_j(\mathbf{x}) \leq 0$ e $h_j(\mathbf{x}) = 0$) para definir a região viável do problema:

$$\text{Minimize} \quad f(\mathbf{x}) \quad (2.3)$$

$$\text{Sujeito a} \quad g_i(\mathbf{x}) \leq 0, i = 1, \dots, m; \quad (2.4)$$

$$h_j(\mathbf{x}) = 0, j = 1, \dots, p; \quad (2.5)$$

$$l_k \leq x_k \leq u_k, k = 1, \dots, r. \quad (2.6)$$

Usualmente, restrições de igualdade são transformadas em inequações adicionando uma pequena tolerância.

Cada problema de otimização possui características próprias, que estarão intimamente relacionadas com a escolha do algoritmo para resolvê-lo. Porém, o ponto inicial é certamente a escolha dos métodos clássicos de otimização, que na maioria das vezes necessitam de informações de derivadas. Por esse motivo, o universo das funções que podem ser otimizadas por eles fica reduzido ao seleto grupo de funções que são contínuas, convexas e semi-modais. Por essa limitação, métodos de otimização que não requerem o uso de derivadas, como é o caso dos algoritmos evolutivos, costumam ser mais populares, sendo essa a principal vantagem de sua utilização.

Algoritmos evolutivos geralmente usam mecanismos inspirados na evolução biológica, como reprodução, mutação, recombinação e seleção para evoluir uma população de soluções em busca da solução ótima de problemas de otimização. Essa busca é guiada por uma função aptidão, que qualifica as soluções e exerce importante papel na etapa de seleção das soluções geradas. Algoritmo genético [1, 27] e evolução diferencial [5] são dois importantes exemplos de algoritmos evolutivos amplamente utilizados na literatura.

2.1 Algoritmos evolutivos em problemas de otimização com restrições

Como já mencionado anteriormente, problemas de otimização com restrições estabelecem um conceito muito importante: a factibilidade das soluções. Uma solução é factível caso ela pertença ao domínio D e satisfaça todas as restrições do problema. Caso contrário, ela é infactível.

Lidar com soluções factíveis e infactíveis no decorrer da evolução quando se aplica um algoritmo evolutivo para resolver problemas de otimização é uma tarefa que não é trivial. Apesar de existir evidências de que as informações provenientes de soluções infactíveis podem ser benéficas para a evolução, tanto em relação à redução do tempo computacional quanto para a obtenção do ótimo global [28, 29], muitos tratamentos de restrições sugerem que elas sejam simplesmente excluídas do processo evolutivo. É o caso, por exemplo, dos métodos de reparo, que foram concebidos para manter apenas soluções factíveis ao longo do processo evolutivo, usando o conhecimento do domínio para mover uma solução infactível para um conjunto factível [30, 31].

Um dos tratamentos de restrição mais populares transforma um problema com restrições em um outro sem restrições somando (ou multiplicando) uma função de penalidade $p(\mathbf{x})$ à função objetivo. Essa operação define uma nova função $F(\mathbf{x})$ chamada de função aptidão (ou *fitness*):

$$F(\mathbf{x}) = f(\mathbf{x}) \diamond p(\mathbf{x}) \quad (2.7)$$

Onde \diamond representa $+$ em penalizações aditivas e \times , em multiplicativas.

A função de penalidade mais simples é a chamada *deathpenalty* [21] que atribui valores arbitrariamente grandes de penalidades ou simplesmente descarta as soluções inviáveis do processo de otimização.

Como dito anteriormente, soluções inviáveis podem fornecer informações importantes na busca pelo ótimo global. Por isso, vários tratamentos de restrições foram concebidos para manter e gerenciar as soluções inviáveis que forem surgindo durante o processo de busca. Um deles, por exemplo, é a técnica de penalização estática clássica, que consiste em representar o termo de penalização $p(\mathbf{x})$ como a soma dos resíduos das violações das equações ou inequações $v_j(\mathbf{x})$ associadas a cada restrição j ponderadas por fatores de penalidade k_j que escalam a importância ou o grau de gravidade de cada restrição. Considerando, por exemplo, as m restrições de desigualdade, tem-se:

$$p(\mathbf{x}) = \sum_{j=1}^m k_j v_j(\mathbf{x}) \quad (2.8)$$

Nas técnicas de penalização estática, os fatores k_j da Equação (2.8) possuem valores previamente definidos pelo usuário e que se mantêm ao longo da evolução do algoritmo. Embora seja de fácil implementação, esta abordagem possui inconvenientes relevantes [21], incluindo a necessidade de um ajuste fino e cuidadoso dos valores para cada k_j que corresponde a uma determinada restrição $v_j(\mathbf{x})$. Este ajuste está fortemente associado ao problema de otimização, o que exige um processo de “tentativa e erro” que pode levar a custos computacionais elevados e pode não ser facilmente generalizado. Ou seja, valores que são adequados para um cenário podem não ser adequados para um outro diferente. Para superar essas questões, foram planejadas abordagens de penalidades dinâmicas e adaptativas [32–34], que consideram a variação dos fatores de penalidades k_j em função do tempo ou da geração, ou coletando informações da população durante o processo de evolução para definir auto-adaptativamente valores “ideais” para k_j .

BARBOSA e LEMONGE [6], por exemplo, propõem o APM (do inglês *Adaptive Penalty Method*), que define os parâmetros de penalidade k_j de maneira adaptativa, calculados a cada geração pela seguinte expressão:

$$k_j = | \langle f(\mathbf{x}) \rangle | \frac{\langle v_j(\mathbf{x}) \rangle}{\sum_{l=1}^{m+p} [\langle v_l(\mathbf{x}) \rangle]^2} \quad (2.9)$$

Na Equação (2.9) as indicações de $\langle \cdot \rangle$ representam uma média sobre a população corrente. Por exemplo, $\langle f(\mathbf{x}) \rangle$ representa a média dos valores da função objetivo considerando todas as soluções na população corrente, enquanto que $\langle v_l(\mathbf{x}) \rangle$ representa a média das violações da solução \mathbf{x} na restrição l do problema.

Enquanto tratamentos clássicos de restrições foram baseados no conceito de combinar os valores da função objetivo e dos valores de violação em um valor único, outros mantêm ambos valores separadamente para decidir sobre quando uma solução é melhor do que outra. Esse é o caso do TSM (*Tournament Selection Method*), método proposto por DEB [11], um dos tratamentos mais populares e efetivos atualmente utilizados. O método ranqueia separadamente as soluções de acordo com os valores da função objetivo ou das violações nas restrições e, ao comparar duas soluções, preferirá aquela mais factível. Em outras palavras, duas soluções são comparadas de acordo com o seguinte critério: (1) uma solução factível é sempre preferida sobre uma infactível; (2) entre duas soluções factíveis, a de melhor valor de função objetivo é preferida; e (3) entre duas soluções infactíveis, a de menor valor de violação nas restrições é preferida.

Outro método que segue essa abordagem em separar os valores produzidos pela função objetivo e pelas restrições é o SR (*Stochastic Ranking*) proposto por RUNARSSON e YAO [7]. Ele balanceia a dominância da função objetivo perante as restrições, comparando soluções adjacentes ora pelo valor da função objetivo ora

pelo somatório dos valores de violação nas restrições. Esta escolha é feita por um parâmetro P_f , especificado previamente pelo usuário. Os mesmos autores, mais tarde, propuseram um cálculo da função de aptidão [8] balanceando o valor da função objetivo e o somatório das violações de acordo com a seguinte expressão:

$$F(\mathbf{x}) = P_f \frac{\text{rank}(f(\mathbf{x})) - 1}{N - 1} + (1 - P_f) \frac{\text{rank}(\sum_{j=1}^{m+p} v_j(\mathbf{x})) - 1}{N - 1} \quad (2.10)$$

onde N representa a quantidade de soluções na população e $\text{rank}(f(\mathbf{x}))$ e $\text{rank}(\sum_{j=1}^m v_j(\mathbf{x}))$ representam, respectivamente, a posição da solução \mathbf{x} em um ranking de acordo com o valor da função objetivo e com o somatório das violações nas restrições. Esse método foi chamado GCR, do inglês *Global Competitive Ranking*.

HO e SHIMIZU [9] propuseram um outro método - HSR (*Ho and Shimizu Ranking*) - que balanceia a função objetivo com os valores de violação nas restrições. Nele, as soluções da população são classificadas segundo três diferentes rankings: o primeiro R_f de acordo com o valor da função objetivo, o segundo R_ϕ de acordo com o somatório das violações e o terceiro R_{Nv} de acordo com a quantidade de restrições violadas. Se uma população possuir apenas soluções infactíveis, o valor da função aptidão será calculado pelo somatório das posições segundo R_ϕ e R_{Nv} . Caso contrário, o algoritmo deve explorar o espaço de busca adicionando o ranking R_f . Portanto, a função *fitness* é calculada por:

$$F = \begin{cases} R_f + 2, & \text{para soluções factíveis} \\ R_f + R_\phi + R_{Nv}, & \text{para soluções infactíveis} \end{cases} \quad (2.11)$$

DE CASTRO RODRIGUES *et al.* [10] propuseram, recentemente, um tratamento que combina todas as metodologias descritas anteriormente. Isto é, separaram os valores da função objetivo dos valores das violações das restrições, preferem soluções factíveis a infactíveis e rejeitam o uso de parâmetros pré-fixados. O BRM (*Balanced Ranking Method*) trata inicialmente duas fileiras separadamente: uma de soluções factíveis e outra de infactíveis. Cada solução de cada fila possui sua função aptidão definida de acordo com sua posição na fila. O ranking relativo às soluções da fileira factível é mantido estático, enquanto a fileira das infactíveis varia baseada em uma expressão de penalização adaptativa. Então, a fileira com soluções infactíveis é fundida à fileira factível também seguindo critérios adaptativos baseados no número de soluções factíveis e infactíveis da população, resultando em uma “fila fundida balanceada”, que é uma ordenação do conjunto de soluções candidatas. No geral, o método compreende uma lógica bastante complexa, que é descrita em detalhes em [10].

Todos os tratamentos mencionados anteriormente podem ser encontrados resumidos e com mais detalhes no Apêndice B ou em [35].

2.2 Evolução assistida por modelos substitutos

A função objetivo (2.1) de um problema de otimização pode ser definida de modo que a avaliação de uma solução candidata por ela demande muito tempo e recurso computacional. Isso pode ser devido, por exemplo, ao uso de simuladores utilizados para a avaliação de soluções que não apresentam soluções analíticas. Em um ambiente como este, avaliar todas as soluções de uma população quando um algoritmo evolutivo está sendo usado pode consumir todo o tempo/recurso computacional disponível para a obtenção da solução ótima.

Em problemas computacionalmente caros, como na otimização de sistemas estruturais para a produção de petróleo *offshore* que utiliza o método dos elementos finitos em suas simulações, o uso de modelos substitutos exerce uma importante função em otimização evolutiva. Usar uma função mais barata para substituir uma função objetivo custosa pode reduzir significativamente o tempo computacional de um algoritmo em busca da solução ótima.

Um tipo de modelo substituto bastante utilizado é um baseado em similaridade [36]. Ele aproxima o valor da função objetivo de uma solução escolhendo as k soluções mais próximas em um banco de dados e calculando a média dos valores de suas funções objetivo ponderados pelas suas distâncias. Essa aproximação é chamada de k -NN (do inglês *k-nearest neighbors*).

O k -NN guarda soluções $(x_j, f(x_j))$ avaliadas pela função original do problema em um banco de dados ordenado. Qualquer solução x_h produzida durante o processo evolutivo é avaliada segundo o k -NN pela seguinte média ponderada:

$$f(x_h) = \frac{\sum_{j=1}^k d(x_h, x_j)^u f(x_j)}{\sum_{j=1}^k d(x_h, x_j)^u} \quad (2.12)$$

onde $d(x_h, x_j)$ é a distância Euclidiana entre x_h e x_j , k é o número de vizinhos e $u = 2$.

Definir o tamanho do banco de dados é uma tarefa difícil. Quando eles são grandes, o processo pode se tornar lento e conter muitas soluções similares. Quando eles são pequenos, eles podem não ser eficientes para aproximar uma solução qualquer do espaço de busca. Uma alternativa é fixar o tamanho do banco de dados e estabelecer um gerenciamento inteligente para a atualização das soluções.

Em relação à dinâmica do banco de dados, a metodologia mais utilizada é aquela que seleciona a melhor solução da população corrente para ser inserida no banco de dados [37–39]. Porém, em [19], preferiu-se selecionar soluções aleatoriamente. Sob o argumento de explorar áreas mais distantes, a pior solução também foi selecionada para ser avaliada pela função exata em [40]. Em [18] a factibilidade das soluções é usada para determinar quais delas serão avaliadas pela função original e, conse-

quentemente, inseridas no banco dados. A metodologia mais utilizada para deletar soluções do banco de dados é aquela em que as piores soluções são selecionadas [41]. Em [19] as soluções mais antigas são substituídas.

É evidente que as soluções do banco de dados devem ser atualizadas durante a evolução já que, para melhor ajustar as aproximações, diferentes soluções na população corrente demandam diferentes soluções no banco de dados. Com esse intuito, essa tese adota um gerenciamento do banco de dados baseado em “mérito” (que será detalhado no Capítulo 4).

Capítulo 3

Um tratamento de restrições baseado em múltiplos rankings

Esta tese propõe um tratamento de restrições baseado em ranking chamado de MCR (do inglês *Multiple Constraint Ranking*). Como alguns tratamentos comentados anteriormente neste texto, o MCR organiza as soluções em filas (ou rankings), cada uma por um critério diferente. A função aptidão das soluções é definida, portanto, pelas posições de cada solução em cada fila, como previamente apresentado em [42] e, posteriormente, em [35] (Apêndice B).

Enquanto HO e SHIMIZU [9] calculam a aptidão das soluções considerando apenas um ranking R_ϕ associado aos valores das violações nas restrições (2.11), o MCR o separa em m outros rankings, um para cada restrição $j = 1, \dots, m$. Dessa maneira, a função aptidão F para cada solução é calculada pela seguinte expressão:

$$F = \begin{cases} R_{Nv} + \sum_{j=1}^m R_\phi^j, & \text{se houver apenas soluções inactivas} \\ R_f + R_{Nv} + \sum_{j=1}^m R_\phi^j, & \text{caso contrário} \end{cases} \quad (3.1)$$

onde R_f , R_{Nv} e R_ϕ^j representam a posição da solução em fileiras de acordo com o valor da função objetivo, do número de restrições violadas e da violação em cada restrição j , respectivamente.

A ideia por trás dessa estratégia é a de permitir uma avaliação adequada para restrições com diferentes ordens de grandeza e/ou unidades. Ela evita que o nível de valores de violação de algumas restrições seja dominado por qualquer restrição que apresente valores com maior ordem de grandeza. Assim, independentemente do conjunto de valores de cada tipo de restrição, todas elas receberão a mesma prioridade e importância na função de penalização de cada indivíduo.

Problemas de otimização cujas restrições possuem diferentes ordens de grandeza/unidades são comuns em diversas áreas da engenharia. É o caso, por exemplo, do problema de otimização de rotas de dutos submarinos, que contém restrições de declividade, avaliadas em graus, e restrições de distância mínima entre os poços de petróleo, calculadas em milhões de dólares [26]. Além disso, podem existir restrições discretas para, por exemplo, posicionar os dutos, e contínuas, para definir o raio mínimo de curvatura em alguns pontos da rota. A abordagem proposta pelo MCR torna-se interessante neste tipo de problema, visto que são desconhecidos os valores mínimos e máximos das restrições que o compõem e, portanto, os procedimentos-padrão para normalização podem não ser aplicáveis.

O comportamento do MCR será ilustrado em um exemplo simples: um problema de minimização com 5 soluções (S_i) e 3 restrições (v_j) de desigualdade. Para cada solução S_i , a Tabela 3.1 apresenta os valores da função objetivo f e das funções de violações v_1 , v_2 e v_3 (com ordens de grandezas notadamente diferentes). Existe apenas uma solução factível na população: S_5 , com valores de violação nulos para todas as restrições.

Tabela 3.1: MCR, exemplo ilustrativo: definição do problema

Solução	f	v_1	v_2	v_3
S_1	5,41	6,24	$3,4 \times 10^5$	0.002
S_2	1,13	7,8	0	0.03
S_3	8,7	3,1	$4,3 \times 10^5$	0.0012
S_4	2	0	0	0.04
S_5	10	0	0	0

Os rankings calculados para cada indivíduo (segundo a função objetivo R_f , número de restrições violadas R_{Nv} e as violações para cada restrição $R_\phi^1, R_\phi^2, R_\phi^3$) são apresentados na Tabela 3.2. Obviamente, os valores dos rankings variam entre 1 e o número de soluções, neste caso $N = 5$. A última coluna indica o valor da função aptidão F , obtida pela soma dos valores das colunas anteriores. Observa-se que o valor do ranking da coluna relativa à função objetivo foi considerado, já que existe uma solução factível (S_5). É possível notar que a melhor solução (isto é, aquela com o menor valor para F) é a S_5 , que é a única solução factível na população. Isso cumpre a suposição comum dos tratamentos de restrições: a de preferir soluções factíveis mesmo que estas apresentem alto valor de função objetivo f . Por outro lado, a segunda melhor solução é a S_4 , que viola apenas uma das restrições e possui o segundo menor valor para f .

Tabela 3.2: MCR, exemplo ilustrativo: resultados

Solução	Ranking					F
	R_f	R_{Nv}	R_ϕ^1	R_ϕ^2	R_ϕ^3	
S_1	3	4	4	4	3	18
S_2	1	3	5	1	4	14
S_3	4	4	3	5	2	18
S_4	2	2	1	1	5	11
S_5	5	1	1	1	1	9

3.1 Experimentos computacionais com o MCR

Inicialmente, o MCR e os outros tratamentos de restrições descritos na Seção 2.1 foram implementados em um algoritmo genético com as seguintes características: representação com codificação real, 100 soluções na população, seleção baseada em ranking, *crossover* BLX- α com $\alpha = 0.15$ e probabilidade igual a 0,9 [43], mutação gaussiana com taxa igual a 0,03 e foi aplicado um esquema de elitismo. Claro que outros algoritmos de otimização poderiam ter sido selecionados, assim como outros parâmetros. Porém estes citados foram fixados para prover um ambiente único de neutralidade para que as comparações fossem feitas da maneira mais imparcial possível. A maioria dos tratamentos não demandam o ajuste de parâmetros pelo usuário, exceto os SR e o GCR que requerem a definição de um valor para o parâmetro de probabilidade P_f . Nas comparações feitas nesse trabalho, foi utilizado $P_f = 0.35$, seguindo recomendações da literatura que alcançaram as melhores soluções para os problemas de otimização utilizados aqui.

Quatro conjuntos de experimentos, resumidos na Tabela 3.3, foram utilizados como experimentos computacionais para este capítulo. Os dois primeiros conjuntos incluem funções de referência em otimização restrita e foram propostos em competições do IEEE-CEC (*Congress on Evolutionary Computation*) para problemas de otimização com parâmetros reais, respectivamente, nas edições de 2006 e 2010. Problemas de otimização que são referências em engenharia compreendem o terceiro e quarto conjunto: o primeiro inclui cinco conhecidos problemas de projeto de engenharia, enquanto que o segundo envolve problemas relacionados à otimização estrutural.

Vinte e cinco execuções independentes do algoritmo foram realizadas sobre cada problema teste utilizando cada um dos tratamentos de restrição. O critério de parada é definido pelo número máximo de avaliações da função objetivo (MaxFEs), que é especificado para cada problema na última coluna da Tabela 3.3. As metodologias utilizadas para comparar os resultados foram: (i) comparação direta entre as melhores soluções; (ii) *Sign Test* [53, 54], que é uma medida estatística não-paramétrica que valida diferenças significativas entre resultados e (iii) perfis de desempenho [55], que é uma ferramenta importante na sumarização dos resultados, auxiliando na

Tabela 3.3: Problemas de otimização: experimentos computacionais

Conjunto	Descrição	Referência	MaxFEs $\times 10^3$
1	Funções CEC 2006	[44]	500
2	Funções CEC 2010	[45]	200
3	Pressure Vessel	[46]	500
	Welded Beam	[47]	500
	Catilever Beam	[48]	500
	Speed Reducer	[49]	500
	Tension/compression Spring	[49]	500
4	10-Bar truss continuous (T10C)	[50]	280
	10-Bar truss discrete (T10D)	[50]	90
	25-Bar truss discrete (T25D)	[51]	20
	52-Bar truss discrete (T52D)	[52]	17,5

visualização deles.

As comparações foram realizadas segundo cada bloco de problemas da Tabela 3.3. Também foram agrupados alguns problemas específicos que possuem restrições com diferentes ordens de magnitude e/ou unidades para verificar se o MCR é, de fato, eficiente neste tipo de problema.

Todos os resultados relacionados a essas comparações foram publicados em [35] e apresentados no Apêndice B. Nesta publicação, análises rigorosamente detalhadas foram feitas e discutidas. Deixaremos a encargo do leitor a leitura deste artigo para que as conclusões descritas na próxima seção sejam validadas.

3.2 Conclusões: MCR

Como dito anteriormente, o MCR é uma técnica de tratamento de restrições baseada em ranking para lidar, principalmente, com restrições caracterizadas por diferenças de ordens de grandezas e/ou unidades. Ele também foi concebido totalmente desacoplado do algoritmo de otimização, permitindo sua associação com qualquer outro algoritmo evolutivo. Além disso, o MCR não requer nenhuma definição prévia de parâmetros pelo usuário, é simples e fácil de se implementar. Uma análise detalhada do seu desempenho é apresentada e comparada com outros seis tratamentos de restrições em [35] (Apêndice B), todos implementados no mesmo algoritmo genético e aplicados aos problemas de otimização restrita da Tabela 3.3.

As comparações gerais indicam o seu bom desempenho, não apenas em termos do número de sucessos na obtenção das soluções ótimas nas comparações diretas entre os tratamentos, mas também por ter sido o tratamento que retornou a melhor solução em maior quantidade de problemas. Embora a robustez (medida pela habilidade de prover mais soluções factíveis) tenha sido ligeiramente superada pelo TSM, o MCR ainda apresentou um desempenho muito bom neste quesito. O desempenho do MCR é significativamente melhorado para a classe de problemas para a qual ele

foi projetado, ou seja, aqueles que apresentam restrições com diferentes magnitudes. Em resumo, o MCR mostrou-se mais preciso e robusto para estes problemas, ao mesmo tempo em que é competitivo para todos os outros conjuntos de problemas. Estes resultados indicam o potencial do MCR para solucionar de maneira eficiente problemas práticos e complexos de engenharia.

Capítulo 4

Um modelo de aproximação baseado em similaridade com gerenciamento do banco de dados via mérito

A utilização do k -NN como função de aproximação demanda a definição de um banco de dados, que armazena soluções avaliadas pela função exata. k dessas soluções são selecionadas sempre que necessário para aproximar soluções da população corrente através da Equação (2.12).

Nesta tese, o gerenciamento do banco de dados (ou a atualização das suas soluções) segue um esquema de mérito [56] onde leva-se em conta a contribuição das soluções no processo de aproximação para decidir quais delas permanecerão e quais serão substituídas. Para avaliar uma solução por essa aproximação, uma ou mais soluções vizinhas são selecionadas do banco de dados para calcular a função de aptidão aproximada. Tais vizinhos recebem “1 ponto” como recompensa, enquanto que os outros (aqueles que não são vizinhos) serão penalizados com a retirada de “1 ponto”. No final de cada geração, uma porcentagem das soluções da população corrente é avaliada pela função exata e introduzida no banco de dados, substituindo as soluções que possuem menos pontos. Esse processo beneficia soluções que são mais similares à população corrente durante todo o processo evolutivo, reduzindo o erro relacionado ao uso do modelo substituto.

Nesse esquema observa-se que três importantes parâmetros são introduzidos: o tamanho do banco de dados, o número k de vizinhos e a porcentagem de soluções selecionadas da população corrente para ingressar no banco de dados. Um estudo desses parâmetros foi feito sobre os mesmos problemas utilizados nesta tese em [56] e a seguinte configuração foi selecionada: tamanho do banco de dados igual a

$1,5N$ (onde N é o número de soluções da população corrente), $k = 2$ e 22% como porcentagem de soluções selecionadas da população corrente para serem avaliadas pela função exata e introduzidas no banco de dados. Esses mesmos parâmetros são utilizados nesta tese.

Para entender o funcionamento deste gerenciamento, um ambiente ilustrativo foi montado considerando 2 soluções na população corrente $S_1 = (0, 0)$ e $S_2 = (6, 5)$ para minimizar uma função objetivo $f(x_1, x_2) = x_1 + x_2$. A Tabela 4.1 mostra em suas colunas, da esquerda para a direita, o processo de aproximação das soluções S_i e de atualização do banco de dados. Na primeira coluna as 4 (apenas neste exemplo será utilizado $2 \times N$ ao invés de $1,5 \times N$) soluções BD_i do banco de dados são avaliadas segundo a função objetivo exata f . Para calcular as aproximações de S_1 e S_2 , as segunda e terceira colunas apresentam as distâncias euclidianas de cada uma em relação às soluções do banco de dados. As $k = 2$ soluções mais próximas de S_1 , BD_3 e BD_4 , são utilizadas para aproximar a função objetivo de S_1 segundo a Equação (2.12) e o valor aproximado é mostrado na quarta coluna. Por terem sido usadas para aproximar $f(S_1)$, ambas soluções BD_3 e BD_4 recebem +1 como pontuação, enquanto que as soluções BD_1 e BD_2 , -1. Este processo se repete para aproximar $f(S_2)$. Quando todas as soluções já tiverem sido aproximadas e todas as pontuações computadas, as pontuações são somadas. A quinta coluna mostra que BD_1 possui a menor pontuação e, portanto, terá prioridade para deixar o banco de dados quando esse for atualizado. Sendo 22% o número de soluções da população corrente selecionadas para ingressarem no banco de dados, uma solução é selecionada segundo o menor valor de f_{approx} , avaliada pela função original f e inserida ao banco de dados em substituição à BD_1 . Como $f(S_1)$ é o menor valor, então S_1 fará parte do novo banco de dados da próxima geração, que está representado na última coluna da tabela.

Tabela 4.1: k NN com gerenciamento do banco de dados por mérito, exemplo ilustrativo: banco de dados definido por 4 soluções (BD_i) avaliadas pela função original (f) para aproximar o valor da função objetivo (f_{approx}) de 2 soluções (S_j) da população corrente, utilizando o sistema de pontuação por mérito para sua atualização

Banco de dados (geração j)	$d(BD_i, S_1)$	$d(BD_i, S_2)$	Aproximação	Pontuação	Banco de dados (geração $j + 1$)
$BD_1 = (9, 8), f = 17$	12,04 (-1)	4,24 (-1)	$f_{approx}(S_1) = 8,81$	-2	$BD_1 = (0, 0), f = 0$
$BD_2 = (5, 5), f = 10$	7,07 (-1)	1 (+1)	$f_{approx}(S_2) = 9,33$	0	$BD_2 = (5, 5), f = 10$
$BD_3 = (5, 4), f = 9$	6,40 (+1)	1,41 (+1)		2	$BD_3 = (5, 4), f = 9$
$BD_4 = (0, 1), f = 1$	1 (+1)	7,21 (-1)		0	$BD_4 = (0, 1), f = 1$

4.1 Experimentos computacionais: SADE-kNN

Esta tese propõe um algoritmo chamado SADE-kNN, definido por um acoplamento entre uma evolução diferencial (DE) e o modelo de aproximação por similaridade k -NN com gerenciamento do banco de dados via mérito.

Primeiramente, o SADE-kNN é comparado com um DE básico, que avalia todas as soluções pela função exata. Depois, ele é comparado com os resultados obtidos pelo algoritmo SA-DECV apresentados em [18] que também acopla um DE a um metamodelo também baseado em similaridade usando o k -NN. Nessa abordagem, ambos os valores da função objetivo e do somatório das violações nas restrições são aproximados. O gerenciamento do banco de dados do SADE-kNN usa a factibilidade das soluções para determinar qual delas serão avaliadas ou pelo modelo exato ou pelo aproximado. O SA-DECV, ao aproximar o valor de violação pelas soluções, também prevê a existência de restrições cujas avaliações sejam dispendiosas computacionalmente. Esta abordagem é interessante, já que nesse caso o algoritmo pode reduzir ainda mais o esforço computacional.

Todos os algoritmos utilizados nas comparações deste capítulo utilizaram o critério de factibilidade proposto por DEB [11] (e explicado no Capítulo 2) como tratamento das restrições dos problemas.

Os experimentos foram executados utilizando 22 funções-teste obtidas de um conjunto de funções propostas em uma das competições do IEEE-CEC em 2006 [44], denominadas G-Suite. Nesta tese não foram consideradas as funções G20 e G22, já que nenhum método obteve solução factível para elas.

O tamanho da população foi fixado em 30 soluções candidatas, valor este comumente utilizado na literatura. De maneira a reduzir o esforço computacional em relação ao número de comparações por similaridade, o número de vizinhos foi fixado em $k = 2$. Os parâmetros do DE foram $F = 0.8$ e $Cr = 90\%$. Estes valores alcançaram boas soluções nos trabalhos [57, 58] para os mesmos problemas abordados nesta tese.

Vinte e cinco execuções independentes foram feitas com 500 mil avaliações da função objetivo em cada uma delas, ambos valores sugeridos em [44]. Foram registrados o número de avaliações necessárias para cada algoritmo obter o ótimo de cada problema, assim como o número de execuções nas quais foram encontradas soluções factíveis. Essas informações nos permitem concluir acerca da economia de avaliações da função objetivo (principal objetivo deste capítulo) e sobre a precisão do algoritmo.

Todos estes resultados foram publicados e analisados em [59] e estão no Apêndice A. Orientamos a leitura deste artigo, já que as conclusões da próxima seção foram embasadas nele.

4.2 Conclusões: KNN com gerenciamento por mérito

Foi possível notar em [59] (Apêndice A) que o SADE-kNN obteve significativa economia de avaliações da função objetivo quando comparado com ambos os algoritmos, o DE básico e o SA-DECV, para a grande maioria dos problemas. Além disso, o algoritmo mostrou-se eficiente em obter tanto soluções factíveis quanto as soluções ótimas dos problemas. Especificamente, foi possível notar que o SADE-kNN obteve significativa redução do número de avaliações da função objetivo para 14 dos 22 problemas quando comparado com o DE básico e em 12 dos 16 problemas, nos quais ambos os algoritmos alcançaram a solução ótima, quando comparado com o algoritmo SA-DECV.

Capítulo 5

O acoplamento de uma evolução diferencial, o MCR e o metamodelo baseado em k NN com gerenciamento via mérito

Nos Capítulos 3 e 4 foram apresentados e analisados, respectivamente, um tratamento de restrições baseado em ranking e um modelo de aproximação de funções baseado em similaridade, cujo banco de dados é atualizado segundo um esquema de mérito. Suas vantagens quando aplicados a problemas com restrições e a problemas que envolvam funções custosas foram analisadas e validadas separadamente, cada um atuando em seus respectivos cenários.

Com o MCR, problemas com restrições foram resolvidos e os resultados foram confrontados com outros relevantes tratamentos de restrições da literatura. Nas análises percebeu-se o bom desempenho do tratamento em resolver problemas com reduzida área de factibilidade. Além disso, o MCR mostrou-se superior aos demais tratamentos quando na presença de restrições com diferentes ordens de magnitude. Por outro lado, o metamodelo baseado em k -NN com atualização do banco de dados baseado em mérito (Algoritmo 1) reduziu a quantidade de avaliações da função objetivo necessárias para se obter a solução ótima na maioria dos problemas. Ele foi confrontado com uma evolução diferencial básica (onde todas as soluções eram avaliadas pela função original do problema) e com os resultados de um artigo recentemente publicado [60].

Contudo, como o principal objetivo desta tese é o de propor uma ferramenta que resolva problemas com restrições e custosos computacionalmente, analisá-las separadamente poderia não ser suficiente para convencer o leitor de que elas também funcionariam bem juntas. Além disso, percebe-se que o MCR e o metamodelo

possuem potencial para atuarem concomitantemente, já que o MCR também evita avaliar a função objetivo quando não existe solução factível na população. Neste caso, o MCR seleciona as soluções de acordo com os rankings relativos às restrições e a quantidade de restrições violadas. Este funcionamento pode ser melhor entendido pelo Algoritmo 2. É possível reparar, por exemplo, na linha 13, que o tratamento de restrições só solicita informações da função objetivo quando ao menos uma das soluções for factível. Caso contrário, a aptidão é calculada por um somatório dos rankings considerando as violações em cada restrição (linhas 5-9) e o número de restrições violadas (linhas 10-12).

O Algoritmo 3 descreve a rotina de uma evolução diferencial assistida pelo k -NN e cuja aptidão é calculada segundo o MCR. As linhas que acoplam o k -NN e o MCR à evolução são, respectivamente, as 23 e 25. Após o algoritmo evoluir uma população, a função objetivo das novas soluções é aproximada segundo o k -NN e, então, o MCR calcula a aptidão de cada dupla de soluções, uma correspondente à solução corrente e uma recém gerada. As linhas finais do Algoritmo 3, 33-34, atualizam o banco de dados selecionando uma porcentagem das soluções da população corrente.

Algoritmo 1: AVALIA A FUNÇÃO OBJETIVO APROXIMADA SEGUNDO k -NN

Entrada: U_t, BD_t
Saída: Função Objetivo (U_t)

```

1 início
2   para ( $i = 1, \dots, NP$ ) faça
3     para ( $j = 1, \dots, 1.5NP$ ) faça
4       | Calcula distância entre  $u_{t,i}$  e  $bd_{t,j}$ 
5     fim
6     Ordena Banco de Dados segundo distância
7     Calcula a função objetivo aproximada  $u_{t,i}$  considerando
       $bd_{t,j}, j = 1, \dots, k$ , por 2.12
8     para ( $j = 1, \dots, k$ ) faça
9       | Pontua  $bd_{t,j}$  com +1
10    fim
11    para ( $j = k + 1, \dots, 1.5NP$ ) faça
12      | Pontua  $bd_{t,j}$  com -1
13    fim
14  fim
15 fim
```

Algoritmo 2: CALCULA A APTIDÃO SEGUNDO O MCR

Entrada: $u_{t,i}, x_{t,i}$
Saída: $fitness(u_{t,i}), fitness(x_{t,i})$

- 1 **início**
- 2 $fitness(u_{t,i}) \leftarrow 0$
- 3 $fitness(x_{t,i}) \leftarrow 0$
- 4 Defina o conjunto $I = \{u_{t,i}, x_{t,i}\}$
- 5 **para** (cada restrição j) **faça**
- 6 Ordena I segundo a violação na restrição j
- 7 $fitness(u_{t,i}) \leftarrow fitness(u_{t,i}) + \text{ranking de } u_{t,i} \text{ em } I$
- 8 $fitness(x_{t,i}) \leftarrow fitness(x_{t,i}) + \text{ranking de } x_{t,i} \text{ em } I$
- 9 **fim**
- 10 Ordena I segundo o número de restrições violadas j
- 11 $fitness(u_{t,i}) \leftarrow fitness(u_{t,i}) + \text{ranking de } u_{t,i} \text{ em } I$
- 12 $fitness(x_{t,i}) \leftarrow fitness(x_{t,i}) + \text{ranking de } x_{t,i} \text{ em } I$
- 13 **se** (*Existe factível em I*) **então**
- 14 Ordena I segundo a função objetivo
- 15 $fitness(u_{t,i}) \leftarrow fitness(u_{t,i}) + \text{ranking de } u_{t,i} \text{ em } I$
- 16 $fitness(x_{t,i}) \leftarrow fitness(x_{t,i}) + \text{ranking de } x_{t,i} \text{ em } I$
- 17 **fim**
- 18 **fim**

Portanto, neste capítulo, serão apresentados e analisados os resultados da implementação deste algoritmo que acopla uma evolução diferencial (como algoritmo de busca), o MCR (como tratamento de restrições) e o k -NN com mérito (para aproximar funções custosas).

Algoritmo 3: EVOLUÇÃO DIFERENCIAL ASSISTIDA POR k -NN

```
1 início
2    $t \leftarrow 1$ 
3   Inicializar Banco de Dados  $BD_t = \{bd_{t,i}; i = 1, 2, \dots, 1.5NP\}$ 
4   Avalia Exato ( $BD_t$ )
5   Ordene Segundo FO ( $BD_t$ )
6   para ( $t = 1, \dots, NP$ ) faça
7      $x_{t,i} = bd_{t,i}$ 
8   fim
9   enquanto (condição de parada não é atendida) faça
10    para ( $i = 1, \dots, NP$ ) faça
11      Selecione aleatoriamente  $r_1, r_2, r_3 \in 1, \dots, NP$ 
12      Selecione aleatoriamente  $\delta_i \in 1, \dots, n$ 
13      para ( $j = 1, \dots, n$ ) faça
14        se ( $N_{[0,1]} \leq CR$  ou  $j = \delta_i$ ) então
15           $u_{t,i,j} = x_{t,r_1} + F(x_{t,r_2} - x_{t,r_3})$ 
16        fim
17      senão
18         $u_{t,i,j} = x_{t,i,j}$ 
19      fim
20    fim
21  fim
22  Defina  $U_t = \{u_{t,i}, \forall i\}$ 
23  Avalia Aproximado segundo  $k$ -NN ( $U_t$ )
24  para ( $i = 1, \dots, NP$ ) faça
25    Calcula Fitness MCR ( $u_{t,i}, x_{t,i}$ )
26    se ( $fitness(u_{t,i}) < fitness(x_{t,i})$ ) então
27       $x_{t+1,i} \leftarrow u_{t,i}$ 
28    fim
29    senão
30       $x_{t+1,i} \leftarrow x_{t,i}$ 
31    fim
32  fim
33  Avalia uma porcentagem da população corrente pela função exata
34  Substitua as soluções com menos pontos de  $BD_t$  pelas soluções que
    foram avaliadas
35   $t \leftarrow t + 1$ 
36 fim
37 fim
```

5.1 Experimentos computacionais

O Algoritmo 3 foi implementado e aplicado às funções-teste G-suíte [44], as mesmas utilizadas nos capítulos anteriores.

Os parâmetros também foram os mesmos do Capítulo 4: 30 soluções na população, taxa de cruzamento de 90% e $F = 0.8$. Em relação aos parâmetros do

metamodelo baseado em k -NN, foi mantido $k = 2$, tamanho do banco de dados $= 1.5 \times NP$ e 22% de cada geração avaliada pela função original do problema para substituir as soluções menos pontuadas do banco de dados. Foram realizadas 25 execuções independentes do algoritmo para cada problema, possibilitando extrair resultados estatísticos importantes, como a execução que encontrou a melhor e a pior solução, assim como a média das execuções. Também foi possível obter informações acerca da quantidade média de avaliações da função objetivo necessárias para a obtenção das soluções ótimas nos problemas.

Em uma análise preliminar foi observado o quanto a evolução diferencial é suscetível à convergência prematura e/ou à estagnação. De fato, alguns trabalhos já vêm sendo publicados com essa preocupação [61, 62]. Em muitos dos problemas, de acordo com a distribuição das soluções no espaço de busca, a evolução convergia rapidamente para um ótimo local. A tarefa de se desvencilhar dele para explorar outras áreas do espaço de busca é muito árdua, sendo comum, nesta situação, o algoritmo terminar a execução neste mesmo valor. Quando isto acontece, muitas avaliações da função objetivo são feitas inutilmente. Dois exemplos desse comportamento puderam ser observados nos problemas G1 e G11, cujos resultados estão apresentados na Tabela 5.1. No problema G1, o algoritmo encontra a solução ótima (-15) em 21 das 25 execuções utilizando, em média, 3783 avaliações da função objetivo nas execuções ótimas (MNA_{otimo}). Trata-se de um valor muito baixo quando comparado com o limite máximo permitido pelo algoritmo de 500 mil avaliações. Nas outras 4 execuções, o algoritmo fica estagnado no ótimo local -12, até alcançar o critério de parada de 500 mil avaliações da função objetivo. Isso fez com que a estatística MNA_{total} , que representa o número médio de avaliações da função objetivo considerando todas as execuções, não representasse com precisão o desempenho do algoritmo. Se por um lado ele encontra a solução ótima com reduzido número de avaliações da função objetivo em muitas execuções, por outro a média do número dessas avaliações se torna alta por ficar estagnado em um ótimo local em apenas 4 execuções. O mesmo acontece com o problema G11. O algoritmo encontra a solução ótima (0.7499) na maioria das execuções, utilizando poucas avaliações da função objetivo, porém fica estagnado no ótimo local 1.0000 nas demais.

Tabela 5.1: Resultados preliminares encontrados pelo Algoritmo 3 nos problemas G1 e G11

Func [ótimo]	Melhor	Média	Pior	MNA_{otimo}	$Exec_{otimo}$	MNA_{total}
G1 [-15,0000]	-15	-14,5999	-12	3783	21/25	83177,82
G11 [0,7499]	0,7499	0,7736	1,0000	2941,5	14/25	221649,3

Em [63] um algoritmo genético foi implementado para resolver problemas reais de otimização de treliças. Neste artigo, o autor propõe um “renascimento” da população

sempre que ela se estagna em uma região próxima do ótimo global. Os resultados mostraram que a reinicialização da população permitiu encontrar melhores estruturas de treliças. Inspirado nesta abordagem, uma estratégia de reinicialização da população é considerada sempre que ela converge para um ótimo local.

5.1.1 SADE- k NN-MCR acoplado a uma estratégia que evita a estagnação em ótimos locais

Ao perceber este comportamento, foi implementada uma estratégia que permite que, em diferentes momentos da evolução, mais áreas do espaço de busca pudessem ser exploradas quando o algoritmo se estagna em um ótimo local por muitas gerações. Trata-se da reinicialização da população de maneira aleatória, porém mantendo o banco de dados da geração anterior. Esse procedimento faz com que a população reinicializada tenda a caminhar para a região antes estagnada, porém, agora, com a oportunidade de visitar novas regiões e ser influenciada por elas. Ao reinicializar a população nenhuma avaliação extra da função objetivo é feita já que, a princípio, todas as novas soluções serão avaliadas pelo k -NN. Portanto, esta estratégia não impacta no número de avaliações da função objetivo. Ao considerar esta estratégia, um novo parâmetro é inserido ao algoritmo: o número de gerações estagnadas para que a população seja reinicializada. Alguns experimentos foram conduzidos e foi percebido que este parâmetro não impacta consideravelmente no resultado final da evolução quando variado. Isso porque as informações mais importantes são preservadas no banco de dados para evoluir a “nova” população, seja ela reinicializada a cada 1000 ou 10000 gerações estagnadas. Dessa maneira, fixou-se este parâmetro em 1000 gerações.

O Algoritmo 3 com reinicialização da população acima descrita será aqui chamado de SADE- k NN-MCR (do inglês, *Surrogate Assisted Differential Evolution coupled with a k -NN and MCR*). A Tabela 5.2 apresenta algumas estatísticas dos resultados obtidos ao aplicar o algoritmo SADE- k NN-MCR os 22 problemas da G Suite. São divulgadas, para cada problema e considerando todas as 25 execuções, a melhor solução, a média, a pior, o número médio de avaliações da função objetivo para alcançar a solução ótima (MNA_{otimo}), o número de execuções que alcançaram o ótimo ($Exec_{otimo}$), o número médio total de avaliações da função objetivo (MNA_{total}) e a factibilidade (número de execuções que obtiveram soluções factíveis).

É imediato perceber a facilidade do algoritmo SADE- k NN-MCR em obter soluções factíveis. Isso pode ser visto pela última coluna da tabela, que mostra uma factibilidade de 25/25 para quase todos os problemas, exceto para a G17 (21/25) e a G21 (23/25). O algoritmo não alcançou o ótimo global para a G17, mas obteve a solução ótima do problema G21 em 15 das 25 execuções. Outro fator relevante

Tabela 5.2: Resultados encontrados pelo SADE- k -NN-MCR para o conjunto de funções teste G-Suíte

Função [ótimo]	Melhor	Média	Pior	$MNA_{ótimo}$	$Exec_{ótimo}$	MNA_{total}	Factibilidade
G1 [-15.0000]	-15	-15	-15	2528.5	25/25	2528.5	25/25
G2 [-0.8036]	-0.8036	-0.7729	-0.7186	28632.7	4/25	424581.2	25/25
G3 [-1.0005]	-1.0005	-0.4379	-0.0487	169089.4	6/25	380873.3	25/25
G4 [-30665.5386]	-30665.5386	-30665.5386	-30665.5386	1208.2	25/25	1208.2	25/25
G5 [5126.4967]	5126.4967	5126.4967	5126.4967	14612.6	25/25	14612.6	25/25
G6 [-6961.8138]	-6961.8138	-6961.8138	-6961.8138	655.5	25/25	655.5	25/25
G7 [24.3062]	24.3062	24.3062	24.3062	64067.2	25/25	64067.2	25/25
G8 [-0.0958]	-0.0958	-0.0958	-0.0958	182.8	25/25	182.8	25/25
G9 [680.6300]	680.6300	680.6300	680.6300	38226.8	25/25	38226.8	25/25
G10 [7049.2480]	7049.2480	7049.2480	7049.2480	20198.4	25/25	20198.4	25/25
G11 [0.7499]	0.7499	0.7499	0.7499	1334.2	25/25	1334.2	25/25
G12 [-1.0000]	-1.0000	-1.0000	-1.0000	211.5	25/25	211.5	25/25
G13 [0.0539]	0.0539	0.0539	0.0539	48682.1	25/25	48682.1	25/25
G14 [-47.7649]	-47.7649	-47.7649	-47.7649	54356.1	25/25	54356.1	25/25
G15 [961.7150]	961.7150	961.7150	961.7150	5412.4	25/25	5412.4	25/25
G16 [-1.9051]	-1.9051	-1.9051	-1.9051	1447.8	25/25	1447.8	25/25
G17 [8853.5339]	8853.5395	8866.0853	8938.5165	-	0/25	500000	21/25
G18 [-0.8660]	-0.8660	-0.8660	-0.8660	5855.1	25/25	5855.1	25/25
G19 [32.6555]	32.6569	32.6586	32.6676	-	0/25	500000	25/25
G21 [193.7245]	193.7245	199.4284	324.7028	122454.0	15/25	258371.4	23/25
G23 [-400.0550]	-400.0550	-400.0550	-400.0550	51922.6	25/25	51922.6	25/25
G24 [-5.5080]	-5.5080	-5.5080	-5.5080	370.0	25/25	370.0	25/25

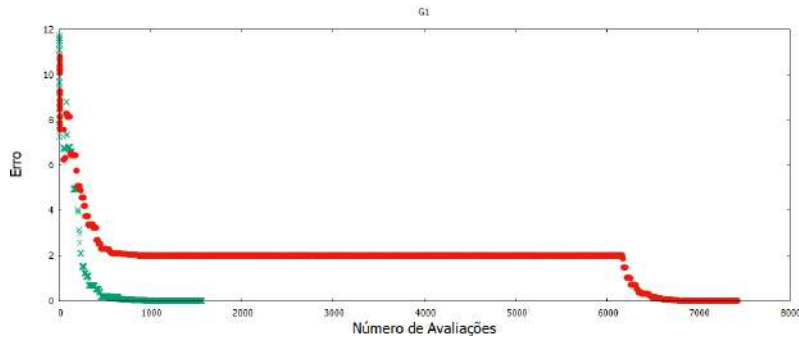
na aplicação do algoritmo SADE-kNN-MCR nestes problemas recai na obtenção da solução ótima. Entre os 22 problemas, o algoritmo só não encontra soluções ótimas para a G17 e a G19, mesmo que suas melhores soluções sejam muito próximas deles. Na maioria dos outros problemas, o algoritmo encontra a solução ótima em todas as execuções.

Mesmo não obtendo a solução ótima em todas as execuções para os problemas G02, G03, G17, G19 e G21, percebe-se que a média das melhores soluções é próxima dela. Ou seja, mesmo não obtendo a solução ótima em algumas execuções, o algoritmo ainda encontra boas soluções. Estas análises qualificam o MCR como um competitivo tratamento de restrições, eficiente na busca de regiões factíveis e ótimas.

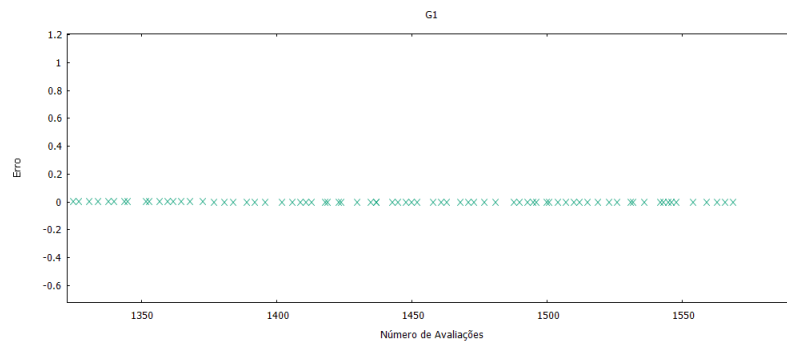
Contudo, a quantidade de avaliações da função objetivo utilizada durante a evolução também é um fator importante. Por isso, a mesma Tabela 5.2 informa o número de avaliações da função objetivo utilizadas para se encontrar o ótimo (MNA_{otimo}) lembrando que, como critério de parada, foi imposto um limite de 500 mil avaliações da função objetivo, como sugerido por [44]. O maior MNA_{otimo} foi 169089,4 (G03), que representa 33% deste número. Os problemas G08, G12, G24 e G06, por exemplo, consumiram, respectivamente, apenas 182,8, 211,5, 370,0 e 655,5 avaliações da função objetivo. Portanto, o algoritmo mostrou-se eficiente tanto em obter as soluções ótimas dos problemas, quanto em diminuir o número de avaliações da função objetivo, principal objetivo desta tese.

Para analisar a evolução da população em alguns problemas, foram plotados gráficos cujo eixo das abscissas representa o número de avaliações da função objetivo e, o eixo das ordenadas, o erro em relação à solução ótima conhecida. A Figura 5.1, mostra, por exemplo, a evolução de duas execuções no problema G01 em que ambas foram conduzidas para o valor ótimo (Figura 5.1a) e as ampliações das curvas nas proximidades do erro igual a zero (Figuras 5.1b e 5.1c). Na evolução relativa à curva verde (cujos pontos foram plotados com “×”) percebe-se na Figura 5.1b que com poucas avaliações (< 800) da função objetivo o erro já assume valores muito próximos a zero. Por outro lado, a execução representada pela curva vermelha mostra uma evolução que consumiu muito mais avaliações da função objetivo para convergir para a solução ótima, como pode ser observado na Figura 5.1c. É possível supor que a população foi reinicializada algumas vezes, já que o erro ficou estagnado em 2 durante muitas avaliações. Nota-se que mesmo quando a população é reinicializada por algumas vezes, não necessariamente ela sairá do ótimo local estagnado. Por outro lado, pode-se dizer que foi a partir de uma reinicialização que a evolução conseguiu se desvencilhar do ótimo local, já com mais de 6 mil avaliações da função objetivo feitas, e convergir para o ótimo global.

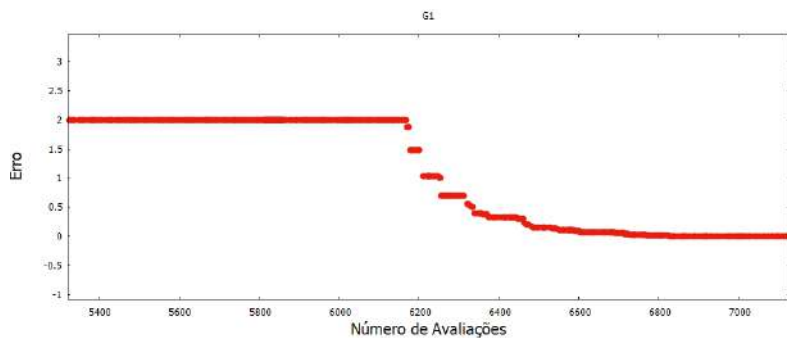
Um outro exemplo interessante são as evoluções de duas execuções no problema



(a) Evolução completa



(b) Ampliação de 5.1a para o final da evolução - convergência sem reinicialização



(c) Ampliação de 5.1a para o final da evolução - convergência com reinicialização

Figura 5.1: Gráfico de evolução para o problema G01 - rodada sem (verde) e com (vermelho) reinicialização da população

G07, plotadas nas Figuras 5.2 e 5.3. A Figura 5.2 mostra uma convergência lenta para o ótimo global. Nesta execução foram utilizadas um pouco mais de 12 mil avaliações da função objetivo sem que houvesse reinicializações da população. O mesmo não acontece com a execução representada pela Figura 5.3. A evolução mostra que a convergência também aconteceu de forma lenta para um valor de erro muito próximo de zero (como pode ser melhor observado pela Figura 5.3c, uma ampliação do gráfico no momento da reinicialização) e, por volta de 23 mil avaliações, a população sofreu uma reinicialização (Figura 5.3b) que se desprende do ótimo local antes estagnado e, a partir daí, conduziu a evolução para o ótimo

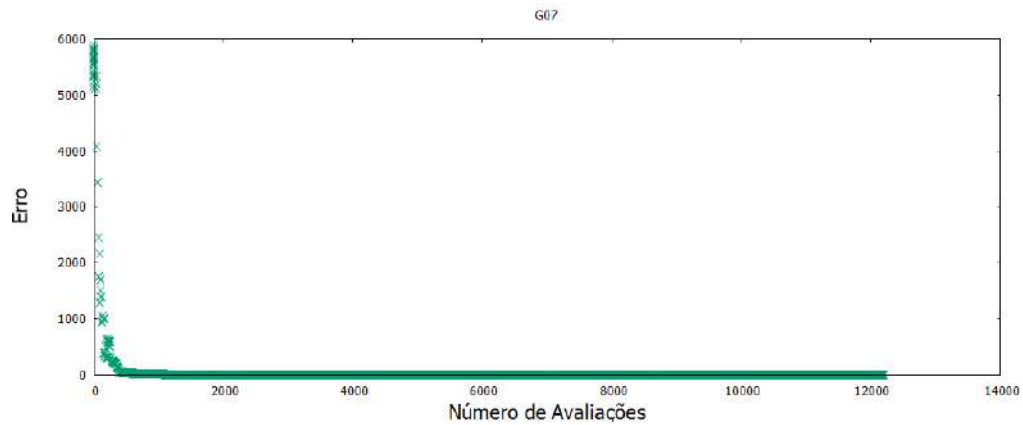
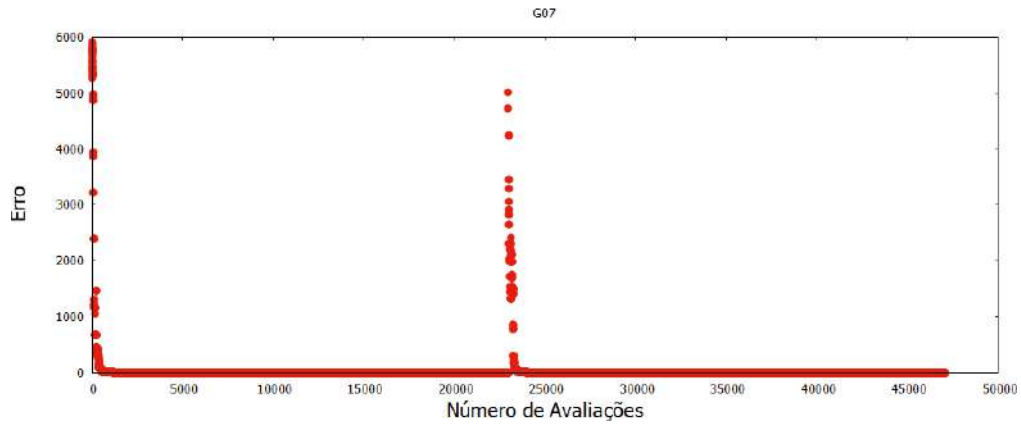


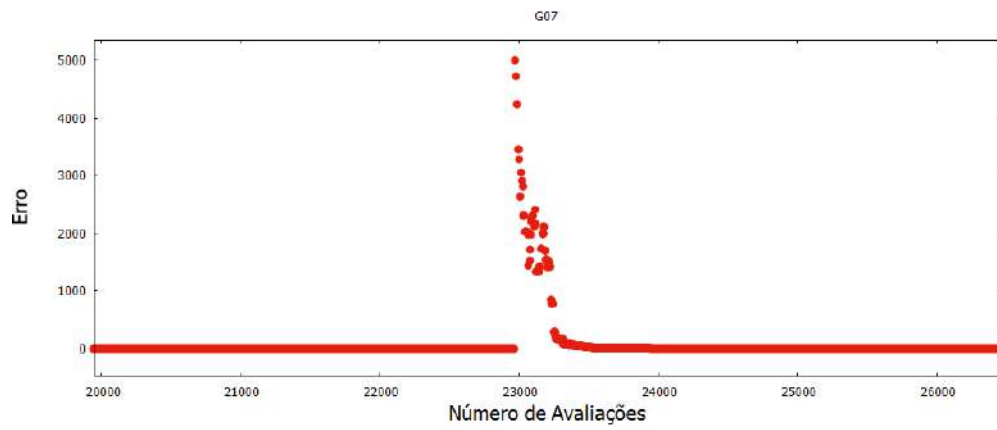
Figura 5.2: Evolução de uma execução ótima sem estagnação para o problema G07

global com um pouco mais de 45 mil avaliações.

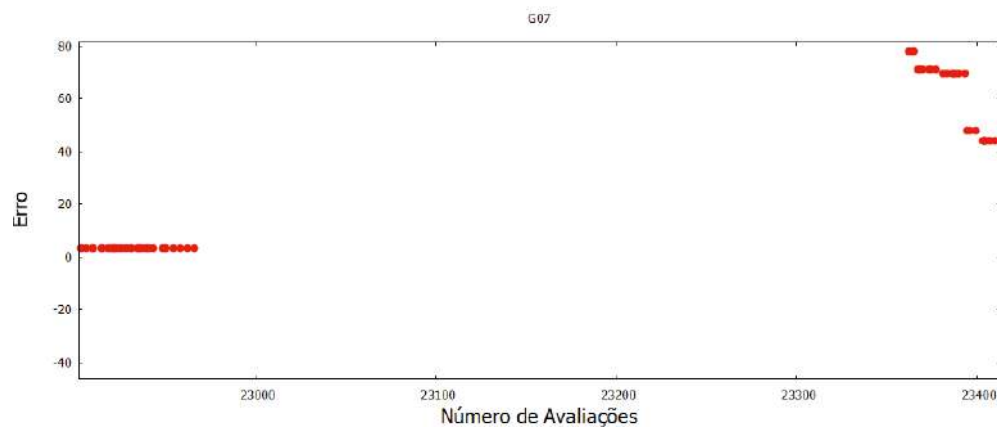
Os comportamentos das evoluções dos únicos dois problemas nos quais as respectivas soluções ótimas não foram encontradas, G17 e G19, podem ser visualizados na Figura 5.4. Percebe-se que, mesmo quando o algoritmo converge para um erro muito próximo a zero, ele não é suficiente para atingir o nível imposto de precisão e, portanto, reinicia a população (aumentando bruscamente o erro) para torná-la a convergir. Estes “picos” de erros se perpetuam durante toda a evolução.



(a) Evolução completa



(b) Comportamento da evolução quando a população é reiniciada

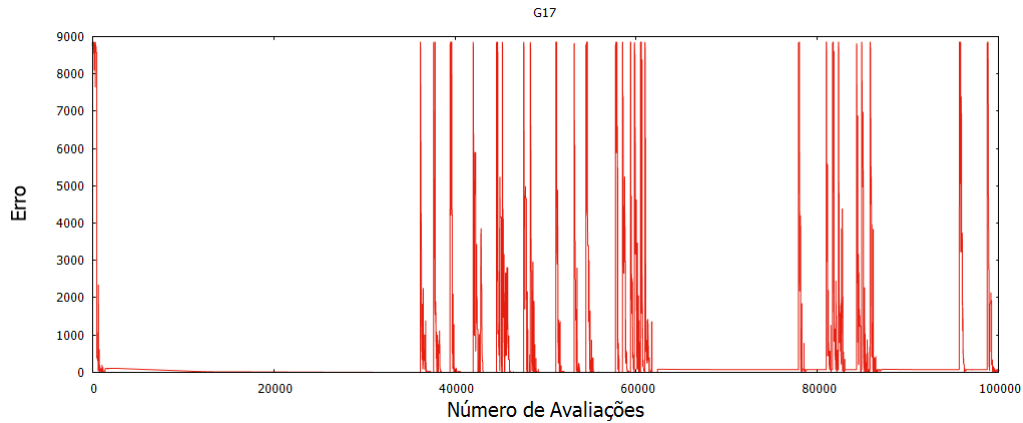


(c) Ampliação da imagem 5.3b no instante da reinicialização

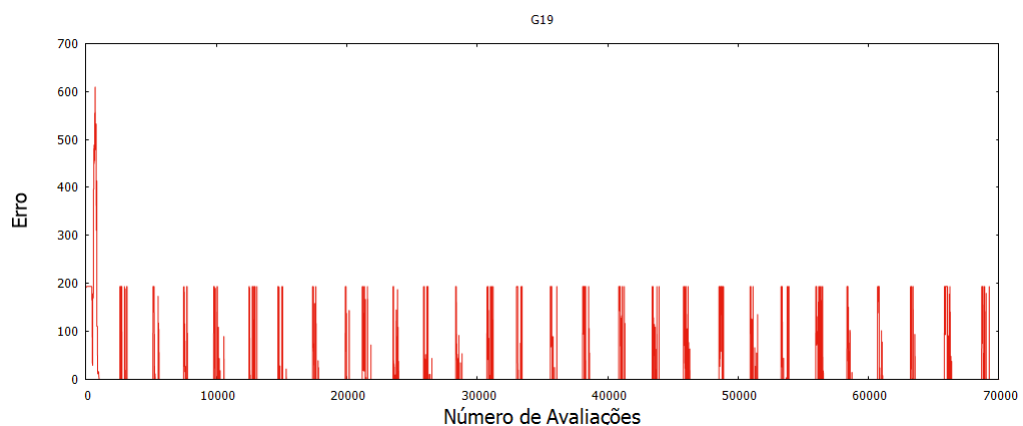
Figura 5.3: Evolução de uma execução com estagnação para o problema G07

5.1.2 Estudo comparativo dos resultados

Nessa seção, os resultados obtidos pelo SADE-kNN-MCR são comparados com os resultados dos algoritmos AG-MCR, apresentado no Capítulo 3 e publicado em [35] (Apêndice B), SADE- k NN, do Capítulo 4 e publicado em [59] (Apêndice A), e e -DE, cujos resultados foram recentemente publicados em [60].



(a) Comportamento da evolução do problema G17



(b) Comportamento da evolução do problema G19

Figura 5.4: Parte da evolução dos problemas G17 e G19

O e -DE é um algoritmo evolutivo baseado na evolução diferencial. A novidade deste algoritmo é que ele adota duas abordagens para cruzamento distintas, sob o argumento de balancear a diversidade e a velocidade de convergência da população, além de adotar sete critérios para comparar soluções factíveis com infactíveis. O algoritmo também é submetido às funções 22 funções do CEC06 [44] e usa os mesmos parâmetros $CR = 0.9$ e $F = 0.8$ utilizados nos experimentos deste capítulo. Além disso, são realizadas 25 execuções independentes mantendo o mesmo nível de precisão $f(x) - f(x^*) \leq 0.0001$. Maiores detalhes sobre este algoritmo podem ser obtidos em [60]. O comportamento em relação à obtenção das soluções ótimas do e -DE e do SADE-kNN-MCR são parecidos. Ambos encontram a solução ótima na maioria dos problemas. Portanto, a comparação entre estes algoritmos só nos interessará em termos do número médio de avaliações da função objetivo necessários para alcançar tais valores ótimos.

A Tabela 5.3 mostra todos os resultados que serão utilizados nesta seção. Na segunda e terceira colunas serão comparadas as melhores soluções obtidas pelo SADE-kNN-MCR e pelo AG-MCR. Na quarta, quinta e sexta colunas os algoritmos SADE-

kNN-MCR, SADE-kNN e e -DE serão comparados em termos do número médio de avaliações da função objetivo necessários para a obtenção da solução ótima. Valores em negrito evidenciam o algoritmo que obteve o melhor resultado.

Percebe-se na segunda e terceira colunas que o SADE-kNN-MCR alcança números melhores para todos os problemas, exceto nos G8, G11 e G12, cujos resultados são iguais. Desconsiderando estes empates, conclui-se que o SADE-kNN-MCR encontra a melhor solução (sendo a maioria ótima) em 19 problemas. Isso representa um p -valor igual a 0.0000019073, o que pode-se dizer que existe uma diferença significativa entre estes dois algoritmos e que o SADE-kNN-MCR possui o melhor desempenho.

Em relação às quantidades médias de avaliações da função objetivo utilizadas para alcançar os valores ótimos dos problemas (quarta, quinta e sexta colunas), percebe-se que o SADE-KNN-MCR utilizou menos avaliações para 15 problemas, enquanto o SADE-kNN e o e -DE em 2 e 5, respectivamente. Em uma comparação direta entre SADE-kNN-MCR e SADE-kNN nota-se que o primeiro utiliza menos avaliações em 19 dos 21 problemas, nos quais ambos encontram a solução ótima. Esses valores correspondem a um p -valor inferior a 0.00012, pequeno suficiente para concluir acerca da diferença significativa entre eles. Já em uma comparação direta com o e -DE, o SADE-kNN-MCR reduz o número médio de avaliações em 15 dos 22 problemas. Estes números representam um p -valor igual a 0.02623, o que permite concluir sobre a diferença significativa entre estes dois algoritmos com 95% de confiança.

Os problemas G10 e G21 possuem restrições com diferentes ordens de magnitudes e foram estudados separadamente em [35] quando o MCR foi apresentado. Como o MCR é um tratamento de restrições desenvolvido especialmente para lidar com este tipo de cenário, uma discussão à parte será feita para estes dois problemas.

A Tabela 5.4 mostra os resultados obtidos pelos algoritmos comparados nesta seção. Percebe-se que apenas o SADE-kNN-MCR e o e -DE encontram as soluções ótimas de ambos os problemas. E, além disso, pode-se perceber, pela Tabela 5.2, que o SADE-kNN-MCR alcança a solução ótima em todas as execuções no problema G10 e em 15 das 25 execuções para o problema G21.

Em relação ao número médio de avaliações da função objetivo, cujos resultados são sumarizados na Tabela 5.5, percebe-se que enquanto o SADE-kNN-MCR reduz o número de avaliações para o problema G10, o e -DE consegue obter a solução ótima do problema G21 com menos avaliações.

Tabela 5.3: Resultados comparativos entre o SADE-kNN-MCR e os algoritmos dos capítulos anteriores no conjunto de funções G Suite

Função [ótimo]	SADE-kNN- MCR Melhor	AG-MCR Melhor	SADE-kNN- MCR MNA_{otimo}	SADE-kNN MNA_{otimo}	e -DE MNA_{otimo}
G1 [-15.0000]	-15	-14.9989	2528.5	3722.4	64274
G2 [-0.8036]	-0.8036	-0.8031	28632.7	-	192297
G3 [-1.0005]	-1.0005	-1.0004	169089.4	-	33066
G4 [-30665.5386]	-30665.5386	-30620.3	1208.2	2597.6	43942
G5 [5126.4967]	5126.4967	5126.56	14612.6	17809.7	152110
G6 [-6961.8138]	-6961.8138	-6431.57	655.5	1234.8	42098
G7 [24.3062]	24.3062	24.9167	64067.2	-	99614
G8 [-0.0958]	-0.0958	-0.0958	182.8	291.7	6254
G9 [680.6300]	680.6300	680.859	38226.8	-	29446
G10 [7049.2480]	7049.2480	7093.59	20198.4	-	135934
G11 [0.7499]	0.7499	0.7499	1334.2	2995.0	24566
G12 [-1.0000]	-1.0000	-1.0000	211.5	385.9	1354
G13 [0.0539]	0.0539	0.4803	48682.1	43906.8	303741
G14 [-47.7649]	-47.7649	-45.7288	54356.1	55179.3	92730
G15 [961.7150]	961.7150	961.73	5412.4	11431.3	95022
G16 [-1.9051]	-1.9051	-1.8417	1447.8	4633.3	24410
G17 [8853.5339]	8853.5395	8862.49	-	69887.3	264232
G18 [-0.8660]	-0.8660	-0.8638	5855.1	253743.7	140458
G19 [32.6555]	32.6569	-	-	-	207286
G21 [193.7245]	193.7245	233.224	122454.0	-	84500
G23 [-400.0550]	-400.0550	-55.4449	51922.6	68852.0	46922
G24 [-5.5080]	-5.5080	-5.5026	370.0	765.2	19678

Tabela 5.4: Comparativo entre as melhores soluções nos problemas cujas restrições apresentam diferenças de magnitudes

Função [ótimo]	SADE-kNN- MCR	AG-MCR	SADE-kNN	e -DE
G10 [7049.2480]	7049.2480	7093.59	7049.249	7049.2480
G21 [193.7245]	193.7245	233.224	193.7546	193.7245

Tabela 5.5: Média do número de avaliações necessárias para alcançar as soluções ótimas em problemas com restrições com diferentes ordens de magnitude

Função	SADE-kNN-MCR	SADE-kNN	e -DE
G10	20198.4	-	135934
G21	122454.0	-	84500

5.2 Conclusões: Evolução diferencial assistida pelo k -NN via mérito e MCR

Neste capítulo foi implementado um algoritmo que acopla uma evolução diferencial, uma aproximação para a função objetivo baseada em similaridade e o MCR como tratamento de restrições. Como alternativa às constantes estagnações em ótimos locais deste algoritmo, foi implementada, em conjunto, uma estratégia que reinicia a população sempre que ela se estagna em um ótimo local por algumas gerações, porém mantendo o banco de dados inalterado. Essa mudança impactou positivamente a busca pelo ótimo global dos problemas e a redução do número médio de avaliações da função objetivo.

Os resultados, apresentados pela Tabela 5.2, mostraram que o algoritmo SADE-kNN-MCR alcança as soluções ótimas para praticamente todos os problemas e em todas as execuções. Além disso, ele é o algoritmo que menos utiliza avaliações da função objetivo quando comparado com os resultados de dois algoritmos desta tese (Capítulos 3 e 4) e um algoritmo também inspirado na evolução diferencial cujos resultados foram recentemente publicados [60].

Quanto aos dois problemas conhecidos por possuírem restrições com diferentes ordens de magnitude, foi possível perceber que o SADE-kNN-MCR alcançou as soluções ótimas em todas as execuções para o problema G10 e em 15 das 25 rodadas para o problema G21. Embora ele tenha reduzido o número médio de avaliações para encontrar a solução ótima do problema G10, o algoritmo e -DE alcança a solução ótima do problema G21 utilizando menos avaliações da função objetivo.

Capítulo 6

Conclusões gerais e trabalhos futuros

O principal objetivo desta tese foi o de propor um algoritmo evolutivo para resolver problemas de otimização definidos por uma função objetivo custosa e cujo domínio de viabilidade das soluções fosse restrito. Assumiu-se, ainda, que tais restrições pudessem ter significativas diferenças de magnitudes entre si quando uma solução fosse avaliada por elas. Então, foi apresentado um tratamento de restrições baseado em ranking para lidar com restrições na solução de problemas de otimização por algoritmos evolutivos. O MCR foi especificamente concebido para solução de problemas de otimização caracterizados por restrições com diferentes ordens de grandeza. Trata-se, também, de uma técnica “desacoplada” do algoritmo de otimização, que não requer a definição de parâmetros, além de sua implementação ser simples.

Na comparação direta do MCR com outros seis tratamentos de restrições [35, 42] observou-se o seu melhor desempenho tanto em número de vitórias diretas entre eles, quanto em relação à sua eficiência em encontrar a melhor solução para mais problemas. Nos problemas cujas restrições apresentavam diferenças de magnitudes, verificou-se que o MCR é significativamente melhor do que os outros tratamentos. Mesmo o MCR tendo sido projetado para resolver problemas cujas restrições apresentam diferenças de magnitudes, ele permanece muito competitivo para todos os outros conjuntos de problemas.

Em relação a problemas definidos por uma função objetivo custosa, foi apresentado um algoritmo de aproximação baseado em similaridade acoplado a uma evolução diferencial para resolver problemas de otimização, chamado de SADE- k NN. As k soluções mais próximas são escolhidas a partir de um banco de dados para contribuir para aproximar soluções da população corrente. O gerenciamento deste banco de dados, onde são armazenadas soluções que foram avaliadas pela função objetivo original do problema, é feito segundo um esquema de mérito. Quanto mais as soluções do banco de dados contribuem para aproximar a função objetivo de

novas soluções, por mais gerações elas permanecem no banco de dados. O SADE- k NN foi comparado com uma evolução diferencial “pura” (onde todas as soluções são avaliadas pela função objetivo original) e com o SA-DECV [18], um algoritmo que também acopla uma evolução diferencial com um k -NN cujo banco de dados é gerenciado de acordo com a factibilidade das soluções.

Os resultados mostraram que o SADE- k NN foi eficiente tanto na obtenção de soluções factíveis e ótimas, quanto em alcançá-las utilizando menos avaliações da função objetivo quando ele foi comparado com a evolução diferencial pura. Quando comparado com o SA-DECV, o SADE- k NN reduziu o número de avaliações da função objetivo para a maioria dos problemas nos quais ambos alcançaram a solução ótima.

Após terem sido apresentados e estudados separadamente, o tratamento de restrição MCR, o algoritmo de aproximação SADE- k NN e uma evolução diferencial foram acoplados para que fossem analisados em conjunto. Também foi considerada uma estratégia que evita a estagnação do algoritmo em ótimos locais. Esta estratégia reinicia a população sempre que ela se estagna por algumas gerações em um ótimo local, porém mantendo inalterado o conjunto de soluções do banco de dados. Este novo algoritmo foi chamado de SADE- k NN-MCR.

Observou-se que o SADE- k NN-MCR obteve as soluções ótimas dos problemas considerados em quase todas as rodadas utilizando, inclusive, menos avaliações da função objetivo quando comparado com outros três algoritmos. Além disso, é possível perceber que ele encontra as soluções ótimas dos problemas que apresentam restrições com diferença de magnitudes com reduzido uso de avaliações da função objetivo.

Os resultados acima discutidos são animadores o suficiente para propor como trabalho futuro a aplicação do algoritmo SADE- k NN-MCR em problemas reais de engenharia. Um deles, por exemplo, é em relação à otimização de *risers* conectados a plataformas flutuantes para produção de petróleo *offshore* [22–24] ou à obtenção de rotas ótimas de gasodutos submarinos [25, 26]. Nesta aplicação, a função objetivo pode ser definida em termos do comprimento do tubo; também, em uma abordagem multi-objetivo, outro objetivo seria maximizar o fluxo, produção e receita de petróleo. Além disso, muitas restrições diferentes são especificadas, como declividade (medida em graus); raio mínimo de curvatura (medido em metros); restrições estruturais, como a estabilidade do fundo e a fadiga devido a vibrações em vãos livres. As restrições podem ser definidas mesmo como quantidades não-dimensionais, como o número de interferências da rota com obstáculos do fundo do mar (equipamentos submarinos, linhas de fluxo, outras tubulações pré-existentes, regiões com corais ou riscos geotécnicos). Para tais problemas, os tratamentos de restrições existentes tendem a priorizar soluções que não violam restrições com maiores mag-

nitudes. Geralmente, mesmo com o uso de tratamento de restrições mais avançados, a influência de restrições com menor magnitude se torna insignificante, uma vez que essas técnicas combinam todas as violações em uma única soma.

Ainda à problemas multi-objetivos, um próximo trabalho seria estudar a eficiência do MCR neste tipo de problema quando o domínio for restrito. Para domínios irrestritos, uma pequena modificação pode ser feita no MCR para considerar cada objetivo uma restrição.

Inspirada no algoritmo SA-DECV, utilizado como comparativo no Capítulo 4, uma próxima abordagem é também considerar o metamodelo baseado no k -NN com gerenciamento do banco de dados por mérito para aproximar o somatório das violações nas restrições para uma parcela das soluções.

Referências Bibliográficas

- [1] HOLLAND, J. H. “Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.” 1975.
- [2] SCHWEFEL, H.-P. “Evolution and optimum seeking. Sixth-generation computer technology series”. 1995.
- [3] KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*, v. 1. MIT press, 1992.
- [4] GOSS, S., ARON, S., DENEUBOURG, J.-L., et al. “Self-organized shortcuts in the Argentine ant”, *Naturwissenschaften*, v. 76, n. 12, pp. 579–581, 1989.
- [5] STORN, R., PRICE, K. “Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces”, *Journal of global optimization*, v. 11, n. 4, pp. 341–359, 1997.
- [6] BARBOSA, H. J., LEMONGE, A. C. “A new adaptive penalty scheme for genetic algorithms”, *Information sciences*, v. 156, n. 3, pp. 215–251, 2003.
- [7] RUNARSSON, T. P., YAO, X. “Stochastic ranking for constrained evolutionary optimization”, *IEEE Transactions on evolutionary computation*, v. 4, n. 3, pp. 284–294, 2000.
- [8] RUNARSSON, T. P., YAO, X. “Continuous selection and self-adaptive evolution strategies”. In: *Evolutionary Computation, 2002. CEC’02. Proceedings of the 2002 Congress on*, v. 1, pp. 279–284. IEEE, 2002.
- [9] HO, P. Y., SHIMIZU, K. “Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme”, *Information Sciences*, v. 177, n. 14, pp. 2985–3004, 2007.
- [10] DE CASTRO RODRIGUES, M., DE LIMA, B. S. L. P., GUIMARÃES, S. “Balanced ranking method for constrained optimization problems using evolutionary algorithms”, *Information Sciences*, v. 327, pp. 71–90, 2016.

- [11] DEB, K. “An efficient constraint handling method for genetic algorithms”, *Computer methods in applied mechanics and engineering*, v. 186, n. 2, pp. 311–338, 2000.
- [12] GREFENSTETTE, J. J., FITZPATRICK, J. M. “Genetic search with approximate function evaluations”. In: *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pp. 112–120, 1985.
- [13] DE PINA, A. C., ALBRECHT, C. H., DE LIMA, B. S. L. P., et al. “Wavelet network meta-models for the analysis of slender offshore structures”, *Engineering Structures*, v. 68, pp. 71–84, 2014.
- [14] DE PINA, A. C., DA FONSECA MONTEIRO, B., ALBRECHT, C. H., et al. “ANN and wavelet network meta-models for the coupled analysis of floating production systems”, *Applied Ocean Research*, v. 48, pp. 21–32, 2014.
- [15] HONG, Y.-S., LEE, H., TAHK, M.-J. “Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks”, *Engineering Optimization*, v. 35, n. 1, pp. 91–102, 2003.
- [16] ANDERSON, K. S., HSU, Y. “Genetic crossover strategy using an approximation concept”. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, v. 1, pp. 527–533. IEEE, 1999.
- [17] RASHEED, K., VATTAM, S., OTHERS. “Comparison of methods for using reduced models to speed up design optimization”. In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pp. 1180–1187. Morgan Kaufmann Publishers Inc., 2002.
- [18] MIRANDA-VARELA, M.-E., MEZURA-MONTES, E. “Surrogate-Assisted Differential Evolution with an Adaptive Evolution Control Based on Feasibility to Solve Constrained Optimization Problems”. In: *Proceedings of Fifth International Conference on Soft Computing for Problem Solving*, pp. 809–822. Springer, 2016.
- [19] FONSECA, L., BARBOSA, H., LEMONGE, A. “A similarity-based surrogate model for enhanced performance in genetic algorithms”, *Opsearch*, v. 46, n. 1, pp. 89–107, 2009.
- [20] LIU, Y., SUN, F. “A fast differential evolution algorithm using k-Nearest Neighbour predictor”, *Expert Systems with Applications*, v. 38, n. 4, pp. 4254–4258, 2011.

- [21] MEZURA-MONTES, E., COELLO, C. A. C. “Constraint-handling in nature-inspired numerical optimization: past, present and future”, *Swarm and Evolutionary Computation*, v. 1, n. 4, pp. 173–194, 2011.
- [22] VIEIRA, I. N., LIMA, B. S. L. P., JACOB, B. P. “Bio-inspired algorithms for the optimization of offshore oil production systems”, *International Journal for Numerical Methods in Engineering*, v. 91, n. 10, pp. 1023–1044, 2012.
- [23] DE LIMA, L. P., DE SOUZA, B., PINHEIRO JACOB, B., et al. “A hybrid fuzzy/genetic algorithm for the design of offshore oil production risers”, *International Journal for numerical methods in engineering*, v. 64, n. 11, pp. 1459–1482, 2005.
- [24] DE PINA, A. A., ALBRECHT, C. H., DE LIMA, B. S. L. P., et al. “Tailoring the particle swarm optimization algorithm for the design of offshore oil production risers”, *Optimization and engineering*, v. 12, n. 1, pp. 215–235, 2011.
- [25] DE LUCENA, R., DE LIMA, B., JACOB, B., et al. “Optimization of pipeline routes using an AIS/adaptive penalty method”. In: *Proceedings of the eighth international conference on engineering computational technology, Civil-Comp Press, Stirlingshire, UK*, 2012.
- [26] DE LUCENA, R. R., BAIOCO, J. S., DE LIMA, B. S. L. P., et al. “Optimal design of submarine pipeline routes by genetic algorithm with different constraint handling techniques”, *Advances in Engineering Software*, v. 76, pp. 110–124, 2014.
- [27] GOLDBERG, D. E. “Genetic algorithms in search, optimization, and machine learning, 1989”, *Reading: Addison-Wesley*, 1989.
- [28] YU, Y., ZHOU, Z.-H. “On the usefulness of infeasible solutions in evolutionary search: A theoretical study”. In: *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 835–840. IEEE, 2008.
- [29] WHILE, L., HINGSTON, P. “Usefulness of infeasible solutions in evolutionary search: An empirical and mathematical study”. In: *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 1363–1370. IEEE, 2013.
- [30] LIEPINS, G. E. “A genetic algorithm approach to multiple-fault diagnosis”, *Handbook of genetic algorithms*, 1991.

- [31] ORVOSH, D., DAVIS, L. “Using a genetic algorithm to optimize problems with feasibility constraints”. In: *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 548–553. IEEE, 1994.
- [32] COELLO, C. A. C. “Use of a self-adaptive penalty approach for engineering optimization problems”, *Computers in Industry*, v. 41, n. 2, pp. 113–127, 2000.
- [33] LIN, C.-Y., WU, W.-H. “Self-organizing adaptive penalty strategy in constrained genetic search”, *Structural and Multidisciplinary Optimization*, v. 26, n. 6, pp. 417–428, 2004.
- [34] WU, W.-H., LIN, C.-Y. “The second generation of self-organizing adaptive penalty strategy for constrained genetic search”, *Advances in Engineering Software*, v. 35, n. 12, pp. 815–825, 2004.
- [35] DE PAULA GARCIA, R., DE LIMA, B. S. L. P., DE CASTRO LEMONGE, A. C., et al. “A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms”, *Computers & Structures*, v. 187, pp. 77–87, 2017.
- [36] SHEPARD, D. “A two-dimensional interpolation function for irregularly-spaced data”. In: *Proceedings of the 1968 23rd ACM national conference*, pp. 517–524. ACM, 1968.
- [37] REGIS, R. G., SHOEMAKER, C. A. “A stochastic radial basis function method for the global optimization of expensive functions”, *INFORMS Journal on Computing*, v. 19, n. 4, pp. 497–509, 2007.
- [38] REGIS, R. G. “Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions”, *IEEE Transactions on Evolutionary Computation*, v. 18, n. 3, pp. 326–347, 2014.
- [39] ELSAYED, S. M., RAY, T., SARKER, R. A. “A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems”. In: *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 1062–1068. IEEE, 2014.
- [40] JIN, Y. “Surrogate-assisted evolutionary computation: Recent advances and future challenges”, *Swarm and Evolutionary Computation*, v. 1, n. 2, pp. 61–70, 2011.

- [41] JIN, Y. “A comprehensive survey of fitness approximation in evolutionary computation”, *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, v. 9, n. 1, pp. 3–12, 2005.
- [42] GARCIA RP, DE LIMA BSLP, J. B. L. A. “Handling optimization problems with constraints of different magnitudes using evolutionary algorithms”. In: *XXXVI Iberian Latin American Congress on Computational Methods in Engineering. Rio de Janeiro*, 2015.
- [43] ESHELMAN, L. J., SCHAFFER, J. D. “Crossover’s niche”. In: *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 9–14. Morgan Kaufmann Publishers Inc., 1993.
- [44] LIANG, J., RUNARSSON, T. P., MEZURA-MONTES, E., et al. “Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization”, *Journal of Applied Mechanics*, v. 41, n. 8, 2006.
- [45] MALLIPEDDI, R., SUGANTHAN, P. N. “Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization”, *Nanyang Technological University, Singapore*, v. 24, 2010.
- [46] SANDGREN, E. “Nonlinear integer and discrete programming in mechanical design optimization”, *Journal of Mechanical Design*, v. 112, n. 2, pp. 223–229, 1990.
- [47] DEB, K. “Optimal design of a welded beam via genetic algorithms”, *AIAA journal*, v. 29, n. 11, pp. 2013–2015, 1991.
- [48] ERBATUR, F., HASANÇEBİ, O., TÜTÜNCÜ, I., et al. “Optimal design of planar and space structures with genetic algorithms”, *Computers & Structures*, v. 75, n. 2, pp. 209–224, 2000.
- [49] 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, pp. 1245–1287, 2002.
- [50] GELLATLY, R. A., BERKE, L. *Optimal structural design*. Relatório técnico, BELL AEROSPACE CO BUFFALO NY, 1971.
- [51] RAJEEV, S., KRISHNAMOORTHY, C. “Discrete optimization of structures using genetic algorithms”, *Journal of structural engineering*, v. 118, n. 5, pp. 1233–1250, 1992.

- [52] WU, S.-J., CHOW, P.-T. “Steady-state genetic algorithms for discrete optimization of trusses”, *Computers & structures*, v. 56, n. 6, pp. 979–991, 1995.
- [53] SHESKIN, D. J. *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.
- [54] DERRAC, J., GARCÍA, S., MOLINA, D., et al. “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, pp. 3–18, 2011.
- [55] DOLAN, E. D., MORÉ, J. J. “Benchmarking optimization software with performance profiles”, *Mathematical programming*, v. 91, n. 2, pp. 201–213, 2002.
- [56] DA SILVA, A. F. *Otimização de problemas com restrição utilizando-se o algoritmo de enxame de partículas assistido por metamodelos*. Tese de Doutorado, Federal Univeristy of Rio de Janeiro, Rio de Janeiro - RJ, 2016.
- [57] GAO, W.-F., YEN, G. G., LIU, S.-Y. “A dual-population differential evolution with coevolution for constrained optimization”, *IEEE transactions on cybernetics*, v. 45, n. 5, pp. 1108–1121, 2015.
- [58] JIA, G., WANG, Y., CAI, Z., et al. “An improved $(\mu + \lambda)$ -constrained differential evolution for constrained optimization”, *Information Sciences*, v. 222, pp. 302–322, 2013.
- [59] DE PAULA GARCIA, R., DE LIMA, B. S. L. P., DE CASTRO LEMONGE, A. C. “A surrogate assisted differential evolution to solve constrained optimization problems”. In: *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pp. 1–6, Nov 2017. doi: 10.1109/LA-CCI.2017.8285681.
- [60] YU, X., LU, Y., WANG, X., et al. “An effective improved differential evolution algorithm to solve constrained optimization problems”, *Soft Computing*, pp. 1–19, 2017.
- [61] LAMPINEN, J., ZELINKA, I., OTHERS. “On stagnation of the differential evolution algorithm”. In: *Proceedings of MENDEL*, pp. 76–83, 2000.
- [62] HRSTKA, O., KUČEROVÁ, A. “Improvements of real coded genetic algorithms based on differential operators preventing premature convergence”, *Advances in Engineering Software*, v. 35, n. 3, pp. 237–246, 2004.

- [63] GALANTE, M. “Genetic algorithms as an approach to optimize real-world trusses”, *International Journal for Numerical Methods in Engineering*, v. 39, n. 3, pp. 361–382, 1996.

Apêndice A

A Surrogate Assisted Differential Evolution to Solve Constrained Optimization Problems

A Surrogate Assisted Differential Evolution to Solve Constrained Optimization Problems

Rafael de Paula Garcia
Civil Engineering Program
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
garcia816@gmail.com

Beatriz Souza L. P. de Lima
Civil Engineering Program
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
bia@coc.ufrj.br

Afonso Celso de Castro Lemonge
Dept. of Applied and Computational
Mechanics
Federal University of Juiz de Fora
Juiz de Fora, Brazil
afonso.lemonge@uff.edu.br

Abstract— Many real optimization problems are defined by functions whose evaluations are very expensive in terms of time consuming. This is an important issue to the applications of evolutionary algorithms, which demand a large number of function evaluations. In this sense, surrogate models can provide good approximations maintaining the accuracy of the search. This paper presents a similarity-based surrogate coupled to a differential evolution to solve constrained optimization problems. The database management, where solutions evaluated by the exact model are stored and used to approximate other solutions, is done according to a “merit” scheme. A set of 24 constrained benchmark problems is used in the computational experiments and the results showed that the proposed scheme achieves good solutions with a reduced number of function evaluations.

Keywords—Surrogate, differential evolution, constrained optimization problems, database management

I. INTRODUCTION

Evolutionary algorithms (EAs) are algorithms which use operators inspired by biological evolution to evolve a population of solutions over generations. Its use in optimization problems became attractive with the introduction of many complex applications in which classical optimization was no longer efficient. This complexity includes high dimensionality problems, non-linear and problem specific constraints and of functions (models) that demand high computational cost to be evaluated. Examples of those types of algorithms are: Genetic Algorithm (GA) [1, 2], Particle Swarm Optimization (PSO) [3], Artificial Immune Systems (AIS) [5, 6] and Differential Evolution (DE) [7].

The application of EAs in constrained problems demands the use of Constraint Handling Techniques (CHTs), which can be based on: (i) Penalization, which adds a penalty value to the objective function of the infeasible solutions [1, 2]; (ii) Rank, which ranks the solutions of the population according to specific criteria [11-13, 4]; and (iii) Selection, which can prioritizes feasible solutions instead of infeasible ones [14].

Evaluating a candidate solution in the optimization process of complex engineering problems is a time and resource consuming task. The use of EAs in such problems can become impractical, since a large number of function evaluations is

required till a satisfactory solution is found. One alternative to deal with these problems is to build a cheaper approximation model (metamodel or surrogate model) to partially substitute the computationally expensive exact functions. Those surrogate models may comprise polynomials, radial basis function, neural networks, and similarity-based models as the nearest-neighbor algorithm. In competitions of the IEEE Congress on Evolutionary Computation (CEC), many papers are using these surrogate models coupled with EAs to deal with computationally expensive functions, and some advances as well as future challenges can be seen in [10].

The foremost application of surrogate models in evolutionary computing was used to approximate the objective function using a polynomial model [29]. In [23] the number of the objective function evaluations is reduced by training a kriging surrogate using a small number of samples evaluated by the expensive function. Surrogate models based on neural networks are used in [20, 22] to reduce the processing time for nonlinear dynamic finite element methods for the analysis of offshore structures. In [21] an online multi-layer neural network approximates the fitness function to decrease the computational time. Other works use surrogates in the initialization of the candidate solutions population, in the evolutionary operators like mutation and crossover [25, 27] or even to approximate the sum of the violations [9].

Few works have studied the efficiency of DE when used in environments that involve approximation functions. In [30], a neural network was used as an approximation technique with a DE and the results showed its good capacity to deal with demanding problems. A similar algorithm was proposed by [31], in which neural networks of radial base functions are employed. In [32] multiple offspring are generated for each parent and the most promising one based on the accuracy and the predicted function value of the current surrogate model is chosen. In [34] and [35] are proposed a local similarity-based surrogate model, based on an r-nearest neighbors, in order to improve DE's overall performance in computationally expensive structural optimization problems.

Surrogate models based on k-nearest-neighbor (k-NN) have been recently used in literature due to its easy implementation and good results [24, 33, 9]. The evaluation of a solution according to the k-NN approximation method may be done by

calculating the average of the objective function of the k -nearest solutions weighted by its distances. Due to this simplicity, this work uses the k -NN approximation as a surrogate model coupled to a DE to solve a set of 24 benchmark problems. The results are compared with those obtained with the same configuration of DE without any surrogate model and with the results presented in [9].

II. DIFFERENTIAL EVOLUTION APPLIED TO CONSTRAINED OPTIMIZATION PROBLEMS

An optimization problem can be described by an n -dimensional space, whose solutions $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ are enclosed by a lower bound (l_k) and an upper bound (u_k) which the aim is minimize or maximize an objective function $f(\mathbf{x})$.

A search space consists of two disjoint subsets, one feasible and another infeasible. The infeasible space is defined by m functions $g_j(\mathbf{x})$ and p functions $h_j(\mathbf{x})$, the inequality and equality constraints, respectively. A candidate solution is considered feasible if it satisfies all constraints. Mathematically, a constrained optimization problem can be written as follows:

$$\begin{aligned} & \text{Min/Max} && f(\mathbf{x}), \\ & \text{Subject to} && g_i(\mathbf{x}) \leq 0, i = 1, \dots, m; \\ & && h_j(\mathbf{x}) = 0, j = 1, \dots, p; \\ & && l_k \leq x_k \leq u_k, k = 1, \dots, n. \end{aligned} \quad (1)$$

Usually, equality constraints are transformed in inequality ones adding a small tolerance, which in this paper is 10^{-4} .

A. Differential Evolution

The differential evolution, proposed by Storn and Price [7, 8], is a population metaheuristic used as a global search algorithm to solve numerical optimization problems. To evolve, evolutionary operators as mutation, crossover and selection are applied upon its solutions.

For each solution x_i , a trial vector u_i is generated using mutation and crossover operators. In the mutation process three random vectors ($x_{r0} \neq x_{r1} \neq x_{r2} \neq x_i$) are chosen to generate the mutant vector v_{ij} .

$$v_{ij} = x_{r0,j} + F(x_{r1,j} - x_{r2,j}), \quad (2)$$

where F is a control amplification typically between 0 and 2.

In the crossover process, the trial vectors are finally generated by the discrete combination of the target and the mutant vector showed below.

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } (rand_j \leq Cr) \text{ or } (j = jrand) \\ x_{i,j}, & \text{otherwise} \end{cases} \quad (3)$$

where Cr is the crossover rate, $rand_j$ is a random number between 0 and 1 and $jrand$ ensures that at least one variable of the mutant vector will be copied into the trial vector.

Finally, a selection operator decides whether the trial vector and the target vector continue to the next generation. In this paper, a pair-wise tournament selection operator proposed by

Deb [14] is used. By this criterion two solutions are compared according to the following:

1. Any feasible solution is preferred to any infeasible solution;
2. Among two feasible solutions, the one having better objective function value is preferred;
3. Among two infeasible solutions, the one having smaller constraint violation is preferred.

III. EVOLUTION ASSISTED BY SURROGATE MODELS

In computationally expensive problems the use of surrogate models plays an important role in evolutionary optimization. Using a cheaper function can significantly reduce time and computational costs. In this perspective, several meta-models are being proposed. Among them, the most used are the polynomial models or response surface models, artificial neural networks and kriging.

Another kind of surrogate models widely used is based on similarity [17]. They are generally used in classification problems, but also produce good results when applied to constrained and unconstrained problems [24]. They approximate the value of the objective function of a solution choosing the k nearest solutions (in this work the Euclidean distance is used) from a database, and calculate the average of its objective function weighted by its distances. This surrogate is called k -NN (k -nearest neighbors).

The k -NN surrogate model stores solutions $(\mathbf{x}, f(\mathbf{x}))$ evaluated by the exact function in a database as an ordered set D . Any solution \mathbf{x}_h produced during the evolutionary process is evaluated according to the k -NN by the following weighed average:

$$f(\mathbf{x}_h) = \frac{\sum_{j=1}^k d(\mathbf{x}_h, \mathbf{x}_j)^u f(\mathbf{x}_j)}{\sum_{j=1}^k d(\mathbf{x}_h, \mathbf{x}_j)^u}, \quad (4)$$

where $d(\mathbf{x}_h, \mathbf{x}_j)$ is the Euclidean distance between \mathbf{x}_h and \mathbf{x}_j , k is the number of neighbor solutions and $u = 2$.

Defining the database size is a difficult task. For extensive databases, the process can become slower and generate many similar individuals compromising the population diversity. Small databases may not be efficient to approximate any kind of solution scattered through the search space. An alternative is to fix the database size and establish a good database management (insertion, maintenance and replacement of the solutions).

Regarding the dynamic of the database, the most used insertion step is the one in which the best solution of the population is chosen and evaluated by the exact model [15, 18, 19]. However, in [24], random individuals are chosen to be evaluated according to the exact model. Under the argument of exploring farther areas, the worst solution is also chosen to be exactly evaluated [10]. In [9] the feasibility of the solutions is used to determine which ones will be evaluated by the exact function. The most widely method used to delete solutions from the database is the one in which solutions with the worst

objective function value are selected [16]. In [24] the oldest solutions are selected to be replaced.

Ideally, the solutions in the database should be updated during the evolution, since different solutions in the current population demand different solutions in the database for better adjusting the approximations. So, we adopt a merit-based account proposed by [26] as an alternative of the dataset management.

A. A merit-based database management

The merit-based management proposed in [26] defines the solutions replacement in the database according to their contribution to the approximation process.

In this process, one or more neighbor solutions are selected from the surrogate model to define the fitness value of the new solution in the current population. Those selected neighbor solutions get "1 point" as a reward while the others (those that are not neighbors) are punished losing "1 point". At the end of each generation, a percentage of solutions of the current population is evaluated by the exact function and introduced into the database replacing the worst solutions which are that with fewer points. This process benefits solutions which are more similar to the current population, decreasing the error related to the use of the surrogate model.

It is observed that three important parameters are introduced: the database size, the number of neighbors k and the percentage of solutions selected in the current population. A parameter study was made in [26] and the following configuration was selected: $|D| = 1.5N$ (where N is the number of solutions of the current population), $k = 2$ and 22% as the percentage of solutions selected from the population, evaluated by the exact function and introduced into the database. These same parameters are used in this work.

IV. COMPUTATIONAL EXPERIMENTS

This paper presents an optimization algorithm, called SADE-kNN, composed by a DE assisted by a surrogate model k-NN, whose database is managed according to the merit scheme.

First SADE-kNN is compared with a "pure" DE which evaluates all solutions by the exact functions. Later, it is compared with the results presented in [9] by the algorithm SA-DECV which couples DE with a surrogate model also based in k-NN. This approach approximates both the value of the objective function and the sum of the constraint violations. Its database management uses the feasibility of solutions to determine which ones will be evaluated either by the exact model or by the surrogate model. All algorithms presented here use the feasibility criteria proposed by Deb [14] presented in section II.

The parameters defined for the computational experiments are: the population size is set equal to 30, commonly used in the literature. In order to reduce the computational effort, with respect to the similarity comparisons, the number of neighbors is set equal to $k = 2$. The parameters of the DE are $F = 0.8$, $Cr = 90\%$, which have achieved good solutions upon those problems [36, 37].

The experiments were executed using twenty two test functions, taken from the well-known G-suite set of benchmark functions [28]. Here we have not considered the functions G20 and G22 for which no method has currently found the optimal solution.

Twenty five independent runs were set with 500,000 function evaluations in each of them, both values suggested in [28]. We also registered the number of function evaluations (NFES) needed by the algorithms to find optimal solutions and the number of runs in which the optimal solution was obtained (NROt). Tables I to IV shows in the first column the known optimal values in brackets. The results are compared in terms of the value of the best solution, and the mean value calculated amongst the optimal solutions provided by the 25 independent runs. The third and fourth columns displays the values related to each algorithm such as: Best, Mean, mean of the NFES and the feasibility rate (number of runs that ended up with at least one feasible solution per total of runs).

Table I presents the results of the problems in which both methods SADE-kNN and DE reached the known global optimum solution. It is noted that there is no significant differences in NROt between them (less than 10%) for most problems, except in G14 and G17, where SADE-kNN obtains the optimum in 3 and 9 more runs than the DE, respectively; and, in problem G13, which DE obtains the optimal solution in 5 more runs than SADE-kNN. The main advantage of the SADE-kNN is evidenced in bold between brackets, showing the reduction tax of the number of function evaluations by the SADE-kNN in the 14 problems displayed in this table. For G5 and G23, for example, SADE-kNN reduces the amount of evaluation in almost 500%. The problem in which this difference is less expressive, G14, SADE-kNN reduces the number of functions evaluations in 82.8%. In 5 problems (G1, G8, G11, G17 and G24) this reduction is around 250% and in 3 of other problems, G6, G13 and G15, the reduction is over 300%.

Table II shows the results of the problems in which both algorithms did not obtain the known optimal solution in 25 runs. SADE-kNN found the best solution for G3 and both algorithms found the same best solution for G21. On the other hand, DE obtained the best mean for G3. In addition, SADE-kNN achieves more feasible runs in both problems, 25 against 24 of DE in G3, and 16 against 8 of DE in G21.

Table III shows the four problems (G2, G7, G9 and G10) where SADE-kNN did not find the optimum solution. DE obtained the optimum in all 25 runs of the problems G7 and G9, in 9 runs of the problem G2 and in 19 runs of the problem G10. The best solutions found by the algorithms for those problems differ only in the third decimal place for most of them. It must be noted that the SADE-kNN returned feasible solutions for all runs in those problems.

Table IV shows the results for G18 and G19 where SADE-kNN did not present a reduction in the NFES. G18 is the only problem in which both methods found the optimal solution and DE needed less evaluations of the objective function than SADE-kNN. It is also observed that SADE-kNN presented difficulties in obtaining the optimal solution, only 10 runs in 25, while the DE succeeded in all of them. In G19 both methods did

not find the optimal solution and, the results are very close. Also, it is noticed that feasible solutions are found in all runs.

TABLE I. SUMMARY OF THE RESULTS OF THE PROBLEMS IN WHICH BOTH METHODS SADE-kNN AND DE GOT THE OPTIMAL SOLUTION

Problem		Optimization algorithms	
		SADE-kNN	DE
G1 [-15.0000]	Mean	-14.5599	-14.7999
	NFES	3722.4 (-256.1%)	13258.6
	NROt	21/25	23/25
G4 [-30665.5386]	Mean	-30665.5386	-30665.5386
	NFES	2597.60 (-188.2%)	7486.80
	NROt	25/25	25/25
G5 [5126.49]	Mean	5126.49	5126.49
	NFES	17809.7 (-497.2%)	106364.3
	NROt	22/25	23/25
G6 [-6961.8138]	Mean	-6961.8138	-6961.8138
	NFES	1234.88 (-331.7%)	5331.60
	NROt	25/25	25/25
G8 [-0.09582]	Mean	-0.09582	-0.09582
	NFES	291.76 (-242.6%)	999.60
	NROt	25/25	25/25
G11 [0.74990]	Mean	0.9695	0.9516
	NFES	2995.00 (-248.3%)	10432.50
	NROt	3/25	4/25
G12 [-1.0000]	Mean	-1.0000	-1.0000
	NFES	385.96 (-103.9%)	787.20
	NROt	25/25	25/25
G13 [0.05394]	Mean	0.3534	0.2540
	NFES	43906.85 (-365.8%)	204540.00
	NROt	7/25	12/25
G14 [-47.764]	Mean	-47.595	-47.672
	NFES	55179.31 (-82.8%)	100871.05
	NROt	22/25	19/25
G15 [961.7150]	Mean	961.7150	961.7150
	NFES	11431.3 (-335.8%)	49820.87
	NROt	23/25	24/25
G16 [-1.9051]	Mean	-1.9051	-1.9051
	NFES	4633.36 (-188.3%)	13362.00
	NROt	25/25	25/25
G17 [8853.53]	Mean	8868.96	8898.38
	NFES	69887.31 (-232.3%)	232293.00
	NROt	19/25	10/25
G23 [-400.055]	Mean	-400.055	-400.055
	NFES	68852.00 (-454.8%)	382020.00
	NROt	1/25	1/25
G24 [-5.5080]	Mean	-5.5080	-5.5080
	NFES	765.20 (-292.6%)	3004.80
	NROt	25/25	25/25

TABLE II. SUMMARY OF THE RESULTS FOR G3 AND G21, WHICH NONE OF THE METHODS OBTAINED THE OPTIMAL SOLUTION

Problem		Optimization algorithms	
		SADE-kNN	DE
G3 [-1.0005]	Best	-0.4515	-0.2890
	Mean	-0.0399	-0.1198
	Feasibility	25/25	24/25
G21 [193.7245]	Best	193.7546	193.7546
	Mean	193.7546	193.7546
	Feasibility	16/25	8/25

TABLE III. SUMMARY OF THE RESULTS OF THE PROBLEMS IN WHICH THE SADE-kNN METHODS HAD DIFFICULTY IN OBTAINING THE OPTIMAL SOLUTION

Problem		Optimization algorithms	
		SADE-kNN	DE
G2 [-0.8036]	Best	-0.7429	-0.8036
	Mean	-0.6367	-0.7871
	Feasibility	25/25	25/25
G7 [24.3062]	Best	24.3073	24.3062
	Mean	28.2727	24.3062
	Feasibility	25/25	25/25
G9 [680.630]	Best	680.638	680.630
	Mean	681.287	680.630
	Feasibility	25/25	25/25
G10 [7049.2480]	Best	7049.249	7049.248
	Mean	7278.785	7251.622
	Feasibility	25/25	25/25

TABLE IV. SUMMARY OF THE RESULTS OF G18 AND G19

Problem		Optimization algorithms	
		SADE-kNN	DE
G18 [-0.8660]	Mean	-0.8654	-0.8660
	NFES	253743.70	57582.00
	NROt	10/25	25/25
	Feasibility	25/25	25/25
G19 [32.6555]	Best	32.6632	32.6569
	Mean	39.2811	33.2691
	Feasibility	25/25	25/25

Finally, we compare our method SADE-kNN with another surrogate assisted algorithm SA-DECV presented in [9].

In [9] the algorithm SA-DECV uses as stopping criterion 240,000 function evaluations. Even using 500,000 function evaluations in our experiments we decided to make the comparison between both algorithms since we present the number of evaluations to find the optimum.

Table V compares the number of evaluations of the algorithms to find the optimal solutions. It must be highlighted that for SADE-kNN this number is the average number of evaluations for the 25 independent runs and for SA-DECV is the lowest number of evaluations among the 30 runs of the algorithm.

In 10 problems of G-suite, both algorithms get the optimal solution (G01, G04, G05, G06, G08, G11, G12, G15, G16 and G24). Among these problems, SADE-kNN uses less evaluations in 8 of them.

It can also be observed in table V that only SA-DECV reaches the optimal solution for G9 and G18, while only SADE-kNN gets the optimal solution for G13, G14, G17 and G23 with average number of evaluations less than the maximum number of evaluations allowed to SA-DECV.

Therefore, it is possible to say that the SADE-kNN reduces the number of evaluations in at least 12 out of 16 problems.

Regarding the other 6 problems in which both algorithms could not find the optimal solution (G2, G3, G7, G10, G19 and G21), SADE-kNN finds better solutions in 5 of them, except G21. Even so, there is no evidence that SA-DECV is better than SADE-kNN to G21 since it used more runs.

TABLE V. COMPARISON OF AVERAGE NUMBER OF FUNCTION EVALUATIONS FOR SADE-kNN AND LOWEST NUMBER OF EVALUATIONS FOR SA-DECV [9]

Problem	SADE-kNN NFES	SA-DECV Lowest NFES
G01	3722.4	6683
G04	2597.6	3801
G05	17809.7	58688
G06	1234.8	6094
G08	291.7	396
G09	-	8377
G11	2995.0	2653
G12	385.9	363
G13	43906.8	-
G14	55179.3	-
G15	11431.3	21915
G16	4633.3	7728
G17	69887.3	-
G18	-	32751
G23	68852.0	-
G24	765.2	2491

V. CONCLUSIONS

This paper presented the SADE-kNN, a new optimization algorithm defined by a coupling of DE and a surrogate model based in k-NN. The database in the surrogate model is managed according to a merit scheme. The SADE-kNN was compared with two other algorithms: a "pure" DE and with the SA-DECV, an algorithm that couples a DE with a k-NN whose database is managed according to the feasibility of the solutions.

The results showed that when the SADE-kNN was compared with the "pure" DE, in addition to being efficient both in obtaining feasible and optimal solutions, produces them using fewer evaluations of the objective function and, consequently, reduced computational cost. When compared with the SA-DECV it was noted that the SADE-kNN saved evaluations of the objective function to at least 12 out of 16 problems in which both algorithms have found the optimal solutions.

ACKNOWLEDGMENT

The authors acknowledge the support of the Brazilian funding agency FAPERJ (grant number E-26/200.314/2016) and CNPq (grant numbers 306069/2016-4 and 305099/2014-0).

REFERENCES

[1] J. H. Holland, "Adaptation in Natural and Artificial Systems", Ann Arbor MI: University of Michigan Press 1975.
 [2] D. E. Goldberg, "Genetic Algorithms in Search", Optimization and Machine Learning. Addison-Wesley, 1989.
 [3] J. Kennedy and R. C. Eberhardt, "Particle Swarm Optimization", IEEE Conference on Neural Networks, p. 1942-1948, 1995.

[4] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization", IEEE Trans Evol Comput, 4:284-94, 2000.
 [5] D. Dasgupta, "Artificial Immune Systems and Their Applications", Springer-Verlag, 1998.
 [6] L. N. de Castro and F. J. V. Von Zuben, "Learning and Optimization Using the Clonal Selection Principle", IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, 2001.
 [7] K. Price and R. Storn, "Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous space". Technical Report, International Computer Science Institute, 1995.
 [8] R. Storn and K Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces", Journal of global optimization 11.4, 341-359, 1997.
 [9] M. E. Miranda-Varela and E. Mezura-Montes, "Surrogate-assisted differential evolution with an adaptive evolution control based on feasibility to solve constrained optimization problems", In: Proceeding of fifth international conference on soft computing for problem solving, 809-822, 2016.
 [10] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges", Swarm and Evolutionary Computation, 1(1):61-70, 2011.
 [11] P. Y. Ho and K. Shimizu, "Evolutionary constrained optimization using and addition of ranking method and a percentage-based tolerance value adjustment scheme", Information Sciences 177, 2985-3004, 2007.
 [12] M. C. Rodrigues, B. S. L. P. de Lima and S. Guimarães, "Balanced ranking method for constrained optimization problems using evolutionary algorithms", Information Sciences 327, 71-90, 2016.
 [13] R. P. Garcia, B. S. L. P. de Lima, A. C. C. Lemonge and B. P. Jacob, "A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms", Computers & Structures 187, 77-87, 2017.
 [14] K. Deb, "An efficient constraint handling method for genetic algorithms", Comput. Methods Appl. Mech. Engrg. 186 (2-4) 311-338, 2000.
 [15] R. G. Regis, "Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions", Evolutionary Computation, IEEE Transactions on, 18(3):326-347, 2014.
 [16] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation", Soft Computing, 9 (1): 3-12, 2005.
 [17] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data", In: Proc. Of the 1968 23rd ACM National Conference, 517-524, New York, NY, USA, ACM Press, 1968.
 [18] R. G. Regis and C. A. Shoemaker, "A stochastic radial basis function method for the global optimization of expensive functions", INFORMS Journal on Computing, 19(4):497-509, 2007.
 [19] S. M. Elsayed, T. Ray and R. A. Sarker, "A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems", Evolutionary Computation (CEC), 2014, IEEE Congress on, 1062-1068, 2014.
 [20] A. C. de Pina, C. H. Albrecht, B. S. L. P. de Lima and B. P. Jacob, "Wavelet network meta-model for analysis of slender offshore structures", Eng. Struct, 2014.
 [21] Y.-S. Hong, H. Lee and M.-J. Tahk, "Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks", Engineering Optimization, 35(1):91-102, 2003.
 [22] A. C. de Pina, B. F. Monteiro, C. H. Albrecht, B. S. L. P. de Lima and B. P. Jacob, "ANN and wavelet network meta-model for the coupled analysis offloating production system", Applied Ocean Research, 2014.
 [23] A. Ratle, "Optimal sampling strategies for learning a fitness model", In: Proceedings of 1999 Congress on Evolutionary Computation, v. 3, 2078-2085, Washington, D.C., 1999.
 [24] L. G. Fonseca, H. J. C. Barbosa and A. C. C. Lemonge, "A similarity-based surrogate model for enhanced performance in genetic algorithms", Opsearch 46.1 89-107, 2009.
 [25] K. Anderson and Y. Hsu, "Genetic crossover strategy using an approximation concept", In: IEEE Congress on Evolutionary Computation, 527-533, Washington, D.C., 1999.

- [26] A. F. da Silva, "Otimização de problemas com restrição utilizando-se o algoritmo de enxame de partículas assistido por metamodelos" (Unpublished doctoral thesis), Federal Universidade of Rio de Janeiro, Rio de Janeiro, Brazil, 2016.
- [27] K. Rasheed, S. Vattam and X. Ni, "Comparison of methods for using reduced models to speed up design optimization", In: Proceeding of genetic and Evolutionary Computation Conference, 1180-1187, New York, NY, 2002.
- [28] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan and C. A. Coello Coello, et al. "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization", Technical report, Sigapore: School of IEEE, Nanyang Technological University, 2006.
- [29] J. J. Grefenstette and J. M. Fitzpatrick, "Genetic search with approximate fitness evaluations", In: Proc. Of the Intl. Conf. on Genetic Algorithms and Their Applications, p. 112-120, Lawrence Erlbaum, 1985b.
- [30] Y. Wang, Y. Shi, B. Yue and H. Teng, "An efficient differential evolution algorithm with approximate fitness functions using neural networks", In: Proceedings of the 2010 international conference on Artificial intelligence and computational intelligence: part II", 334-341, Berlin, 2010.
- [31] U. Pahner and K. Hameyer, "Adaptive coupling of differential evolution and multiquadrics approximation for the tuning of the optimization process", IEEE Transactions on Magnetics, 36(4): 347-367, 2000.
- [32] J. Zhang and A. C. Sanderson, "DE-AEC: A differential evolution algorithm based on adaptive evolution control", In: Proc. Of the Congress on Evolutionary Computation – CEC", 3824-3830, IEEE, 2007.
- [33] Y. Liu and F. Sun, "A fast differential evolution algorithm using k-nearest neighbour predictor", Expert Systems with Applications, 38(4): 4254-4258, 2011.
- [34] E. Krempser, H. S. Bernardino, H. Barbosa and A. C. C. Lemonge, "Differential evolution assisted by surrogate models for structural optimization problems", In Proceedings of the international conference on computational structures technology (CST), Civil-Comp Press (vol. 49), 2012.
- [35] E. Krempser, H. S. Bernardino, H. Barbosa and A. C. C. Lemonge, "Performance evaluation of local surrogate models in differential evolution based optimum design of truss structures", Engineering Computations, 34(2): 499-547, 2017.
- [36] W. F. Gao, G. G. Yen and S. Y. Liu, "A dual-population differential evolution with coevolution for constrained optimization", *IEEE transactions on cybernetics*, 45(5), 1108-1121, 2015.
- [37] G. Jia, Y. Wang, Z. Cai, and Y. Jin, "An improved $(\mu+\lambda)$ -constrained differential evolution for constrained optimization," *Inform. Sci.*, vol. 222, no. 10, pp. 302–322, Feb. 2013.

Apêndice B

A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms



Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms



Rafael de Paula Garcia^b, Beatriz Souza Leite Pires de Lima^{b,*}, Afonso Celso de Castro Lemonge^c, Breno Pinheiro Jacob^{a,b}

^a LAMCSO – Laboratory of Computer Methods and Offshore Systems¹, Avenida Pedro Calmon, S/N, Cidade Universitária, Ilha do Fundão, Caixa Postal 68.506, 21941-596 Rio de Janeiro, RJ, Brazil

^b PEC/COPPE/UFRJ – Civil Engineering Program, Post-Graduate Institute of the Federal University of Rio de Janeiro, Avenida Pedro Calmon, S/N, Cidade Universitária, Ilha do Fundão, Caixa Postal 68.506, 21941-596 Rio de Janeiro, RJ, Brazil²

^c UFJF – Federal University of Juiz de Fora, Applied and Computational Mechanics Dept., Rua José Lourenço Kelmer, S/n, 36036-330 Juiz de Fora, MG, Brazil³

ARTICLE INFO

Article history:

Received 26 September 2016

Accepted 31 March 2017

Keywords:

Optimization

Genetic algorithms

Constraint-handling techniques

ABSTRACT

This work presents a constraint handling technique (CHT) for the solution of real-world engineering optimization problems by evolutionary algorithms. Referred to as the Multiple Constraint Ranking (MCR), it extends the rank-based approach from many CHTs, by building multiple separate queues based on the values of the objective function and the violation of each constraint. This way, it overcomes difficulties found by other techniques when faced with complex problems characterized by several constraints with different orders of magnitude and/or different units.

The MCR follows an “uncoupled” approach where the CHT is not embedded into the optimization algorithm. Extensive studies are performed to assess its accuracy and robustness, compared to six other up-to-date CHTs, all implemented into the same canonical Genetic Algorithm to allow a neutral and unbiased evaluation. The numerical experiments comprise benchmark functions from the IEEE-CEC competitions on constrained optimization, and also classical structural engineering problems. The performance of the CHTs is compared using efficiency measures in terms of nonparametric statistical tests. The results indicate that the MCR is remarkably more accurate and robust for the subset of problems presenting different-magnitude constraints, while remaining very competitive and one of the top-performers for all other benchmark problems comprising the case studies.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

The use of Evolutionary Algorithms (EAs) and other heuristic methods has been quite common in industry, comprising an efficient alternative for the solution of several types of engineering optimization problems [1–9]. One of the most widely acknowledged EA is the well-known Genetic Algorithm (GA) [10,11]; other methods have also been proposed, such as the Particle Swarm Optimization (PSO) [12,13], Artificial Immune Systems (AIS) [14,15], Ant Colony Optimization (ACO) [16], Crow Search algorithm [17], and many others.

Although originally designed to deal with unconstrained search spaces [18,19], EAs have been successfully complemented by constraint-handling techniques (CHTs) to solve constrained problems, guiding the search process to feasible regions and ideally providing solutions that do not violate any constraint [20–22]. One important line of research consists in studying rank-based CHTs. In this context, Runarsson and Yao [23] proposed the Stochastic Ranking (SR) technique that balances objective and penalty functions by a parameter P_f , producing a ranking by comparing adjacent individuals. Later the same authors proposed the Global Competitive Ranking (GCR) [24] which still balances objective and violation function with the parameter P_f , and defines a fitness function depending of two rankings: according to objective function and the sum of violations. Ho and Shimizu [25] proposed a ranking scheme where the individuals are sorted using a function defined according to three rankings, respectively the objective function; the constraint violation values, and the number of violated constraints. More recently the Balanced Ranking Method

* Corresponding author.

E-mail addresses: garcia816@gmail.com (R.P. Garcia), bia@coc.ufjf.br (Beatriz Souza Leite Pires de Lima), afonso.lemonge@ufjf.br (Afonso Celso de Castro Lemonge), breno@lamcso.coppe.ufjf.br (B.P. Jacob).

¹ <http://www.lamcso.coppe.ufjf.br>.

² <http://www.coc.ufjf.br>.

³ <http://www.ufjf.br/mac/>.

(BRM) [26] was proposed where a merged row is built from two rankings: one for feasible and another for infeasible solutions.

All those methods have undoubtedly presented many advantages over earlier CHTs such as the standard static penalization method, aiming to bypass their inherent shortcomings [20] when applied to real-world, complex engineering problems. However, for such problems there are still some issues and challenges related to the definition of the fitness of a given candidate solution. Currently, the main issue might be how to combine into one term all values involved in the evaluation and comparison of the individuals from a given population, i.e. the objective and violation functions that may have different orders of magnitude and/or different units. This would not be an issue for the most usual cases where the ranges of minimum and maximum violation values are known a priori; these cases could be easily dealt with by usual normalization procedures.

However, there are many real-world engineering applications for which such information is not available and cannot be estimated in advance, including for instance practical applications related to offshore structures such as the optimization of risers connected to floating platforms for offshore oil production [9,27,28], or the optimization of subsea pipeline routes [29,30]. For instance, in this latter application the objective function may be defined in terms of the pipeline length; also, in a multi-objective approach another goal would be to maximize oil flow, production and revenue. Moreover, many disparate constraints are specified, such as declivity (measured in degrees); minimum radius of curvature (in meters); structural constraints such as on-bottom stability [31] and fatigue due to vortex-induced vibrations (VIV) in free spans [32]. Constraints may be defined even as non-dimensional quantities, such as the number of identified interferences of the route with seabed obstacles (subsea equipment, flowlines, other pre-existent pipelines, regions with corals or geotechnical hazards). For such problems, existing CHTs would tend to prioritize solutions-individuals that do not violate constraints with higher magnitudes; eventually, even with the use of more advanced CHTs (such as the GCR or BRM) the influence of constraints with lower magnitude may become insignificant, since those techniques merge all violations into a single sum.

It might be argued that the range of violation values can be estimated by inspecting the search space and evaluating candidate solutions prior, or during, the evolutionary optimization process. However, this might involve some drawbacks. Poor estimations might lead to loss of information, compromising the adequate representation of the constraints; many solutions might be located near or beyond the extremes of the estimated range. Additional evaluations might be required to produce reasonable estimations; this might considerably increase the computational costs (due to the complexity of the analysis methods required to assess the structural constraints). Also, this procedure might also require user intervention, which would not be much convenient. Ultimately, for more accurate estimations the optimization process should be run again.

In this context, this work describes a new ranking-based constrained handling technique, referred here as the Multiple Constraint Ranking (MCR). This method is specifically devised to handle constraints with different orders of magnitude and/or different units, without additional computational overhead associated to the estimation of the range of violation values, and without user intervention. To obtain another desirable characteristic of a CHT, i.e. versatility, the MCR follows a so-called “uncoupled” approach where the CHT is not embedded into the optimization algorithm. This allows its implementation along with different evolutionary algorithms, such as in [33] where an ensemble of four well-known uncoupled CHTs was proposed, each with its own population.

Extensive studies on the MCR are presented, by applying it to several benchmark functions (including those from the IEEE competitions on real parameter constrained optimization, and also classical structural engineering problems), and comparing its performance with other up-to-date CHTs: the Adaptive Penalty Method (APM) [34,35], Tournament Selection Method (TSM) [36], Stochastic Ranking (SR) [23], Global Competitive Ranking (GCR) [24], Ho and Shimizu ranking (HSR) [25] and Balanced Ranking method (BRM) [26]. All CHTs were implemented into the same evolutionary algorithm, thus providing a unique environment that allows a fair, neutral and unbiased evaluation of the efficiency of each CHT, and to compare their efficiency without being influenced by the performance of the optimization algorithm. The comparisons are made using efficiency measures in terms of nonparametric statistical tests.

This paper is organized as follows: Initially, Section 2 summarizes the main concepts related to constrained optimization with evolutionary algorithms, and presents a brief description of the compared CHTs. Section 3 presents the MCR and illustrates its main characteristics by a simple example. The full sets of numerical experiments are described in Section 4, followed by an overall assessment of the results in Section 5, while Section 6 presents a critical analysis for each specific set of experiments. Final remarks and conclusions are presented in Section 7.

2. Constrained optimization with evolutionary algorithms

A general constrained optimization problem may be formally defined by considering a r -dimensional search space comprised by a vector of design variables $\mathbf{x} = (x_1, x_2, x_3, \dots, x_r)$, with components x_i presenting lower and upper bounds $[l_k, u_k]$. The goal is to minimize an objective function $f(\mathbf{x})$, considering inequality and equality constraints (respectively $g_j(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) = 0$) that define the feasible region:

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m \\ & && h_j(\mathbf{x}) = 0, \quad j = 1, \dots, p \\ & && l_k \leq x_k \leq u_k, \quad k = 1, \dots, q \end{aligned} \quad (1)$$

Repair methods have been devised to keep only feasible candidate solutions (FCS) along the evolutionary process, using domain knowledge to move an infeasible offspring into the feasible set [37,38]. One of the most popular approaches to treat constraints has been to transform a constrained optimization problem into an unconstrained one, by adding penalty functions $p(\mathbf{x})$ to the objective function $f(\mathbf{x})$ whenever any given constraint is violated, thus leading to an “expanded” objective function $F(\mathbf{x})$, usually referred simply as the “fitness” function:

$$F(\mathbf{x}) = f(\mathbf{x}) + p(\mathbf{x}) \quad (2)$$

The simplest penalty function $p(\mathbf{x})$ is the so-called “death-penalty” [20] that assigns arbitrarily large penalty values, or simply discards the infeasible candidate solutions (ICS) from the optimization process. However, this would prevent the search from using valuable information from the infeasible solutions; thus, several CHTs have been devised to maintain and manage the ICS that unavoidably arise along the search process. The classical *static penalty* technique consists of representing the penalty term $p(\mathbf{x})$ as the sum of values for violation functions $v_j(\mathbf{x})$ associated to each constraint, proportional to the degree of violation, and affected by positive constants – the *penalty factors* k_j that scale and/or weight the relative importance, or degree of severity, of the constraints. Considering for instance the m inequality constraints g , we have:

$$p(\mathbf{x}) = \sum_{j=1}^m k_j v_j(\mathbf{x}) \quad (3)$$

Here, to assess the relative performance of the MCR, we will consider only more recent CHTs that are able to manage the infeasible solutions. The remainder of this section will briefly describe these competing CHTs.

2.1. Adaptive Penalty Method (APM)

In the static penalty technique, the factors k_i have user-specified values that remain fixed along the evolution of the algorithm. Although easy to implement, this approach has important drawbacks [20], including the need of a careful fine-tuning of the value for each factor k_i that corresponds to a given violation function $v_j(\mathbf{x})$. This tuning is heavily associated to the particular application, requiring a “trial and error” process that may lead to high computational costs, and may not be easily generalized; that is, values that are suitable for one scenario may not be adequate for a different one. To overcome those issues, dynamic and adaptive penalty approaches have been devised [39–41], to consider the variation of the penalty factors k_i as a function of time or a generation counter, or by gathering information from the population during the evolution process to self-adaptively define “ideal” values k_i for each constraint.

In this context, the APM – Adaptive Penalty Method proposed by Barbosa and Lemonge [34,35] allows the variation of the factors k_j , adaptively computed at each generation by the following expression:

$$k_j = |f(\mathbf{x})| \frac{\langle v_j(\mathbf{x}) \rangle}{\sum_{l=1}^m \langle v_l(\mathbf{x}) \rangle^2} \quad (4)$$

In this expression, the angle brackets $\langle \cdot \rangle$ indicate the average over the current population: $\langle f(\mathbf{x}) \rangle$ is the average of the objective function values for all solutions in the current population, and $\langle v_l(\mathbf{x}) \rangle$ is the violation of the l th constraint averaged over the population. This expression leads to higher penalty factors being assigned to the constraints more hardly satisfied, that is, with higher average violation values.

The objective function for each candidate solution is evaluated by the following expression, where an individual is marked as infeasible if at least one of the “hard” criteria is violated:

$$F(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if not violated} \\ f^*(\mathbf{x}) + \sum_{j=1}^m k_j v_j(\mathbf{x}) & \text{otherwise} \end{cases} \quad (5)$$

where the term f^* is given by:

$$f^*(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } f(\mathbf{x}) > \langle f(\mathbf{x}) \rangle \\ \langle f(\mathbf{x}) \rangle & \text{otherwise} \end{cases} \quad (6)$$

2.2. Tournament Selection Method (TSM)

While classical CHTs had been based on the concept of combining the values of the objective function and constraints into a single value, as indicated by Eqs. (2) and (5), other methods maintain those values apart along the evolution process. This is the case of the method proposed by Deb [36], one of the most popular and effective CHTs in current use [20]. Different names have been used to refer to that method, such as *Superiority of Feasible Solutions* (SF) [33] or *Preferring Feasible Solutions* [26]; here it will be referred to as the TSM – Tournament Selection Method.

The TSM ranks separately the solutions according to their objective function values or constraint violation values. It comprises a binary tournament selection, i.e. a pairwise comparison that basically prefers a FCS to any ICS. Each set of two individuals are com-

pared according to the following criteria: (1) a feasible solution (FCS) is always preferred over an infeasible one; (2) between two feasible solutions, the one having the best objective function value is preferred; and (3) between two infeasible solutions (ICS), the one having the lowest constraint violation value is preferred.

2.3. Stochastic Ranking (SR)

Another method that follows this approach of separating the objective function and constraint values is the Stochastic Ranking (SR) method, introduced by Runarsson and Yao [23]. The SR is intended to balance the dominance of the objective and constraint functions. Each individual is evaluated through a stochastic ranking procedure similar to a bubble sort, in which the individuals are compared only to the adjacent neighborhoods; it does not observe the population as a whole. The comparison criterion may be based either on the objective function or the sum of constraint violations, and this choice is randomly determined by a user-specified probability parameter P_f . In other words, given any pair of two adjacent individuals, the probability of their comparison according to the objective function is 1 if both of them are feasible. Otherwise, it is P_f . Despite the improvement in search performance, this approach is sensitive to the parameter P_f and, indeed, the authors suggested a range for this parameter based on a set of experiments.

2.4. Global Competitive Ranking (GCR)

Later, the same authors of the SR method proposed another ranking-based CHT, comprising an improvement of their original formulation. This method, referred as the GCR (Global Competitive Ranking) [24], balances the objective function and the sum of constraints violations according to the following expression:

$$F(x) = P_f \frac{\text{rank}(f(x)) - 1}{N - 1} + (1 - P_f) \frac{\text{rank}\left(\sum_{j=1}^m v_j(x)\right) - 1}{N - 1} \quad (7)$$

In this expression N is the amount of individuals in the population; $\text{rank}(f(x))$ and $\text{rank}\left(\sum_{j=1}^m v_j(x)\right)$ represent the current ranking position of the candidate solution x based on its objective value and the sum of its constraint violations, respectively. In this scheme, an individual is ranked by comparing it with all other members of the population. P_f is a user-defined parameter that specifies the level of penalization given to an infeasible solution, thus also representing a balance parameter. The authors suggest that P_f should be given in the range between 0 and 0.5.

2.5. Ho and Shimizu Ranking (HSR)

Ho and Shimizu [25] proposed another ranking-based technique that balances the objective function against the constraint violations; differently from the previous methods, it does not require any user-defined additional parameters. The individuals of a population are sorted using three ranks. The first rank R_f compares the value of the objective function f ; the second rank R_ϕ compares the squared sum of constraint violations $v_j(\mathbf{x})$; and the third rank R_{Nv} compares the number of constraints violated. The position of a given individual A in each one of these ranks is equal to $1 + d$, where d is the number of individuals that dominate A . Therefore, each one of these three ranks ranges from 1 to N , where N is the size of population.

If a given population has only infeasible individuals, the final value of the objective function will be defined as the sum of the rank terms R_ϕ and R_{Nv} since the goal is to seek the first feasible solution from the search space and the information from f becomes unimportant:

$$F = R_\phi + R_{Nv} \quad (8)$$

Otherwise, if the population has both feasible and infeasible individuals, the algorithm should explore the search space adding the R_f rank to find the optimal solution. Therefore the fitness function is given by:

$$F = \begin{cases} R_f + 2 & \text{for feasible solutions} \\ R_f + R_\phi + R_{Nv} & \text{for infeasible solutions} \end{cases} \quad (9)$$

where R_ϕ and R_{Nv} serve as penalty terms for infeasible solutions; for feasible solutions $R_\phi = R_{Nv} = 1$.

2.6. Balanced Ranking Method (BRM)

The BRM, proposed by Rodrigues et al. [26], follows and combines some of the approaches incorporated in the aforementioned methods, i.e.: separation of objective function and constraint values; preference of a FCS over any ICS; incorporation of adaptive concepts, and absence of adjustable parameters.

The BRM handles two queues in parallel, respectively for the FCS and ICS. Initially the FCS and ICS sets are ranked separately; each FCS and ICS has its fitness set to its ranking position, that is, a number between 1 and the FCS set size or ICS set size. The FCS rank queue is kept static, while the ICS rank queue varies based on an adaptive expression for the penalty function. Then, the ICS queue is merged into the FCS queue, also following adaptive criteria based on the number of feasible and infeasible solutions in the population, resulting in a “balanced merged queue” (MQ) that is an ordered set of candidate solutions. Overall, the BRM comprises a rather complex logic, which is described in detail in [26].

3. Multiple Constraint Ranking technique (MCR)

Again following the main approach of ranking the candidate solutions by taking separate queues (based on the objective function and constraint values), the MCR generalizes these ideas by defining *multiple* ranks, as briefly outlined in [42]. For instance, while the HSR method employs only one rank R_ϕ associated to the violation values (built by a single summation of the values for the functions $v_j(\mathbf{x})$ of all constraints), the MCR splits the violation rank into m other ranks, one for each constraint $j = 1, \dots, m$. Thus, the fitness function F for each individual is evaluated according to the following expression:

$$F = \begin{cases} R_{Nv} + \sum_{j=1}^m R_\phi^j & \text{if only infeasible individuals} \\ R_f + R_{Nv} + \sum_{j=1}^m R_\phi^j & \text{otherwise} \end{cases} \quad (10)$$

The idea behind this strategy is to allow an adequate assessment of constraints with different orders of magnitude and/or different units. It avoids the rank of constraint violation values to be dominated by any given constraint for which the violation function presents values with higher order of magnitude (recalling that, for many practical applications, the possible range of minimum and maximum violation values is not known, so that standard normalization procedures cannot be applied). Thus, irrespective of the range of values for each type of constraint, all are given the same priority and importance in the assessment of each individual.

It is interesting to remark that the MCR may also avoid the computational costs associated to the evaluations of the objective function f when there are no feasible individuals in the population. In those cases, the value of F depends only on the ranks associated to the constraints (R_{Nv} and R_ϕ^j). It can also be observed that the MCR follows a relatively simple logic, being easy to implement.

The behavior of the MCR will now be illustrated by its application to a simple example, comprising a minimization problem with five individuals and three inequality constraints. For each individ-

Table 1

MCR, illustrative example: problem definition.

Individual	f	v_1	v_2	v_3
I1	5.41	6.24	3.4×10^5	0.002
I2	1.13	7.8	0	0.03
I3	8.7	3.1	4.3×10^5	0.0012
I4	2	0	0	0.04
I5	10	0	0	0

Table 2

MCR, illustrative example: results.

Individual	Rank					F
	R_f	R_{Nv}	R_ϕ^1	R_ϕ^2	R_ϕ^3	
I1	3	4	4	4	3	18
I2	1	3	5	1	4	14
I3	4	4	3	5	2	18
I4	2	2	1	1	5	11
I5	5	1	1	1	1	9

ual, **Table 1** presents the values of the objective function f and the violation functions v_1 , v_2 and v_3 (with markedly different orders of magnitude). There is only one feasible individual in this population: I5, with violation values equal to zero for all constraints.

The ranks calculated for those individuals (according to objective values R_f ; number of constraints violated R_{Nv} , violation values for in each constraint R_ϕ^1 , R_ϕ^2 , R_ϕ^3) are presented in **Table 2**. Obviously, the rank values range between 1 and the number of individuals $N = 5$. The last column indicates the final value of their fitness function F . Considering that this population has a feasible individual, the F values are calculated using the second expression of Eq. (10), i.e. adding all ranks from the previous columns; the maximum value for F would be $N(2 + m) = 25$ (otherwise, if there were no feasible individuals, the maximum value would be $N(1 + m) = 20$). One can notice that the best individual (i.e. with the lowest F value) is I5 that is the only feasible individual; this complies with the usual assumption of CHTs to prefer feasible solutions (even though presenting higher values for the objective function f). On the other hand, the second best individual is I4 that violates only one constraint, and has the second lowest value for f .

4. Experiments

The MCR and the other CHTs described in Section 2 have been implemented along with a canonical Genetic Algorithm (GA) with the following characteristics: real-coding representation; population size of 100 individuals; rank-based selection; BLX- α crossover with $\alpha = 0.15$ and probability equal to 0.9 [43]; Gaussian mutation with rate equal to 0.03; employing an elitist scheme. Of course other optimization algorithms could have been selected; but, as mentioned in the Introduction, we have chosen this simple, canonical GA to provide a single environment for a neutral evaluation and comparison of the efficiency of each CHT. Most of the considered CHTs do not have user-adjustable parameters, with the exception of the SR and GCR methods that require the definition of a value for the probability parameter P_f . Here we have selected the value $P_f = 0.35$ following guidelines found in the literature to provide the best results for most of the benchmark problems that will be studied.

Four sets of experiments were executed, summarized in **Table 3** along with their respective test problems. The first two sets include the suites of benchmark functions proposed for the IEEE-CEC competitions on real parameter constrained optimization: respectively the twenty-four functions presented in the CEC 2006 competition

Table 3
Computational experiments.

Set	Description	Ref.	MaxFEs × 10 ³
1	CEC 2006 functions	[44]	500
2	CEC 2010 functions	[45]	200
3	Pressure vessel	[46]	500
	Welded beam	[47]	500
	Cantilever beam	[48]	500
	Speed reducer	[22]	500
	Tension/compression spring	[22]	500
4	10-Bar truss continuous (T10C)	[49]	280
	10-Bar truss discrete (T10D)	[49]	90
	25-Bar truss discrete (T25D)	[50]	20
	52-Bar truss discrete (T52D)	[51]	17.5

[44] and the eighteen functions from CEC 2010 [45], all with 10 dimensions. Their complete formulation and mathematical definition will not be reproduced here, since they are rather lengthy and are already well documented in the literature (see for instance the references cited in Table 3). The third and fourth sets comprise engineering benchmark problems: the former includes five well known structural engineering design problems (Pressure Vessel, Welded Beam, Cantilever Beam, Tension/Compression Spring, and Speed Reducer). Finally, the fourth set comprises standard benchmarks related to the structural optimization of trusses. The formulation of the engineering problems will be presented in Appendix A.

Twenty-five independent executions of the GA were performed for each test problem and each CHT. The termination criterion is defined in terms of the maximum number of function evaluations (MaxFE), which is specified for each problem in the last column of Table 3. Section 5 will present an overall comparison of the performance of all methods in all sets: firstly, directly in terms of the number of problems for which each one found the best solution; then, by using efficiency measures in terms of nonparametric statistical tests, specifically the Sign Test (ST) [52,53], and the Performance Profiles (PPs) [54].

Specific comparisons for each set of experiments will be presented later in Section 6. Particularly, Section 6.5 will group, amongst the test problems included in the four sets of Table 3, those representative of the engineering problems that are the focus of this work, i.e. presenting constraints with different orders of magnitude and/or different units. Lower and upper bounds can be analytically defined for the constraint functions of some of those problems (functions G10 and G21 from the CEC 2006 suite, and the Pressure Vessel problem). The Welded Beam problem presents severely nonlinear constraint functions, thus the orders of magni-

Table 4
Ranges for the constraints with different magnitudes.

G10			G21		
-0.95	$\leq g_1 \leq$	4	-1000	$\leq g_1 \leq$	640.227
-3.45	$\leq g_2 \leq$	3.975	-48,250	$\leq h_1 \leq$	52,500
-10.9	$\leq g_3 \leq$	8.9	-47,625	$\leq h_2 \leq$	39,448
10065000.0	$\leq g_4 \leq$	1748999.187	-0.303071	$\leq h_3 \leq$	0.384611
-11,227,500	$\leq g_5 \leq$	11,227,500	-0.4085	$\leq h_4 \leq$	0.497
-11,240,000	$\leq g_6 \leq$	11,215,000	-1.6449	$\leq h_5 \leq$	1.7146
Pressure vessel			Welded beam (orders of magnitude)		
-3.85375	$\leq g_1 \leq$	4.9807	-10 ⁴	$\leq g_1 \leq$	10 ⁴
-1.90175	$\leq g_2 \leq$	4.9046	-10 ⁴	$\leq g_2 \leq$	10 ⁶
-1288673.3	$\leq g_3 \leq$	57,317,600	-10	$\leq g_3 \leq$	10
40	$\leq g_4 \leq$	230	-10	$\leq g_4 \leq$	10 ²
			-10	$\leq g_5 \leq$	10 ⁻¹
			-10 ⁻¹	$\leq g_6 \leq$	10 ⁴
			-10 ⁷	$\leq g_7 \leq$	10 ³

Table 5
Number of problems for which each CHT found the best solution.

CHT	Set of experiments				Total
	CEC 2006	CEC 2010	Eng. Problems	Trusses	
MCR	8	5	3	2	18
BRM	7	5	2	1	15
TSM	4	4	1	0	9
HSR	3	4	1	1	9
GCR	6	2	1	0	9
APM	4	2	2	0	8
SR	2	1	1	0	4

tude of their bounds have been numerically estimated. These lower and upper bounds are presented in Table 4, where the differences in scale and magnitude are evident. Finally, this group includes three of the truss problems (T10C, T10D and T25D) that have constraints defined both in terms of displacements and stresses, naturally with different units/magnitudes.

5. Overall comparisons

5.1. Direct comparison: number of wins

Table 5 reports the amount of problems for which each algorithm found the best solution. This direct comparison indicates the good performance of the MCR: it outperforms the other methods for all sets of experiments, with the exception of the CEC 2010 suite of problems for which the BRM presented the higher number of wins. The last column of Table 5 shows that, when totalizing the results of all experiments, the MCR provided the overall best results. It is interesting to note that the sum of wins not always corresponds to the total number of problems, since a given problem may be counted more than once when a draw occurs, or may be discarded when neither of the methods returns a feasible solution.

5.2. Sign test

The Sign Test (ST) is a statistical procedure that compares the performance of two methods, and indicates if their performance is statistically different. It is based on pairwise comparisons that counts the number of cases on which a method is the winner. When a method provides a better result for a given problem, it receives a positive sign, while the other receives a negative sign. The significance of the difference between these pairwise comparisons is tested; a significance level is obtained by calculating probability values (*p*-values) [55] from the totalized signs and

Table 6
Sign test, pairwise comparisons.

MCR	BRM		HSR		SR		GCR		APM		TSM	
	#	p-value	#	p-value	#	p-value	#	p-value	#	p-value	#	p-value
Wins	28	0.2879	28	0.2399	38	0.0001	29	0.1264	34	0.0076	34	0.0120
Losses	23		22		12		20		16		17	

comparing them to a threshold value α . If the p -value is higher than the threshold, then the null hypothesis (H_0) that both methods are equivalent is accepted; otherwise the p -value is considered statistically significant and H_0 is rejected, or in other words the performance of the methods is different. In practice, the most commonly used threshold values are 0.01 and 0.05 [53], representing respectively 1% and 5% chance of rejecting the null hypothesis when in fact it is correct.

Table 6 presents the results of the Sign Test based on the pairwise comparison between the MCR and each other CHT. The number of wins and losses confirms the observations based on Table 5, i.e. the MCR outperforms all other methods. Regarding the statistical assessment in terms of the p -values: the highlighted values are below the threshold of 0.05, indicating that the superiority of the MCR over the SR, APM and TSM methods is statistically significant.

5.3. Performance profiles (PPs)

Performance profiles (PPs) [56,57] were introduced in [54] to evaluate and compare the performance of a set of solvers S for a given set of optimization problems P . To assemble a PP, firstly one must select a given metric $m_{p,s}$ to assess each solver: for instance, the number of evaluations of the objective function (as indicated in [54]), or other performance measures such as statistical parameters computed from objective function values obtained in several runs of the algorithm (i.e. mean, standard deviation, best value, etc.), as suggested in [57]. Then, the performance ratio $r_{p,s}$, for solver $s \in S$ applied to the problem $p \in P$ is calculated by dividing this metric $m_{p,s}$ by the corresponding value of the metric obtained by the solver with best performance on this problem:

$$r_{p,s} = \frac{m_{p,s}}{\min\{m_{p,s} : s \in S\}}$$

Obviously, the solver that performs best for a given problem p will have $r_{p,s} = 1$. The ratio $r_{p,s}$ may thus be assumed to be in the range $[1, r_M]$; an arbitrarily large value may be specified for the upper limit r_M , being assigned to $r_{p,s}$ only when solver s is not able to find a solution for problem p . Then, taking n_p as the total number of problems in set P , an overall assessment of the performance of solver s is given by:

$$\rho_s(\tau) = \frac{\text{size}\{p \in P : r_{p,s} \leq \tau\}}{n_p}$$

In this expression, $\rho_s(\tau) \in \mathfrak{R} \rightarrow [0, 1]$ may be seen as the cumulative distribution function for the performance ratio $r_{p,s}$; i.e. the probability for solver s that its ratio $r_{p,s}$ is within a factor τ of the best possible ratio. For $\tau = 1$, $\rho_s(1)$ indicates the accuracy of solver s , i.e. a normalized measure of the number of times it was able to provide the best value for the selected metric, or the probability that it will win over all the other solvers in set S ; $\rho_s(1) = 1$ would indicate that this solver wins over all the other solvers in set S in all problems P . On the other hand, for larger values of τ , $\rho_s(\tau)$ indicates the robustness of the solver, i.e. the number of times it is able to provide feasible solutions: recalling the definition of the upper limit r_M presented above, $\rho_s(r_M) = 1$ and the probability that method s solves a problem is given by $\rho_s^* = \lim_{\tau \rightarrow r_M} \rho_s(\tau)$.

Table 7
Performance profiles (PPs), illustrative example: computational costs $c_{p,s}$.

Algorithm	Problem				
	p_1	p_2	p_3	p_4	p_5
s_1	1.0	1.0	1.0	5.0	3.0
s_2	–	5.5	5.5	1.0	1.0
s_3	2.0	4.0	4.0	6.5	8.0

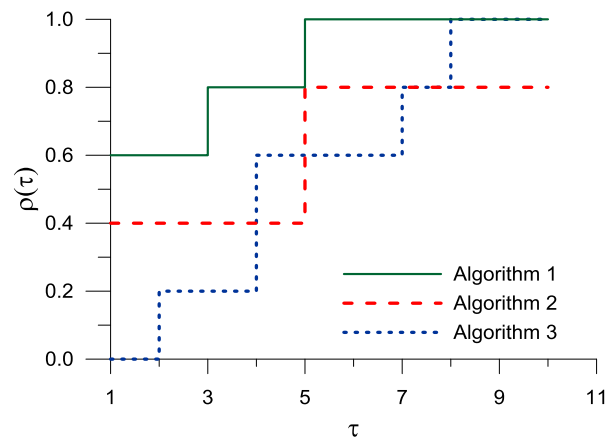


Fig. 1. Performance profiles (PPs), Illustrative Example.

The PPs are comprised by graphs $[\rho_s(\tau), \tau]$ with several curves, one for each solver s . The curves are nondecreasing, piecewise constant functions, continuous from the right at each discontinuity point. An illustrative example of the assembly of PPs is presented in Table 7 and Fig. 1 [56], the selected metric being “normalized” computational costs $c_{p,s}$. The maximum performance ratio is 8.0, so the graph is plotted up to $\tau_{\max} = r_M = 11$. Algorithm s_1 wins for 3 out of 5 problems; this is reflected in its value $\rho_1(1) = 0.6$. The discontinuity points seen in its curve correspond to $\tau = 3$ (when it is outperformed by s_2 in problem p_5 with a ratio $r_{5,1} = 3$), and $\tau = 5$ (outperformed in problem p_4 with a ratio $r_{4,1} = 5$). Algorithms s_1 and s_3 provided solutions for all problems, so their curves both reach the maximum value $\rho_s^* = 1$ for $\tau = \tau_{\max}$; on the other hand, algorithm s_2 did not reach a solution for one of the problems, so its curve stopped at $\rho_s^* = 0.8$. Comparing s_1 and s_3 , the former is the most robust, since it is the first to reach $\rho_s^* = 1$ (for $\tau = 5$). Therefore, in this example it is clear that s_1 is the best algorithm.

The PPs are thus an interesting tool for the easy visualization and interpretation of results from computational experiments. If one is interested only in the number of wins, it is enough to compare the values of $\rho_s(1)$ for all methods; if one is interested in methods with a high probability of success, then the final section of the curves should be investigated, corresponding to $\tau \in [r_s, r_M]$ for some $r_s < r_M$. Overall, the highest the ρ_s values, the better the method is, so a quick visual assessment of the performance of all methods may be performed simply by observing which one has the curves above all others. Conversely, a more precise numerical indication may be provided by calculating the areas under the curves.

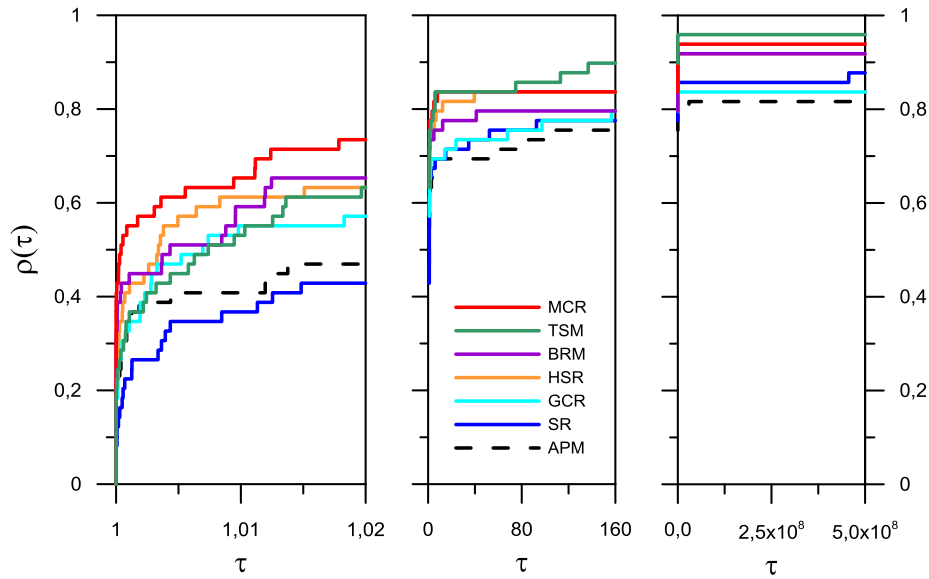


Fig. 2. PPs for all test problems.

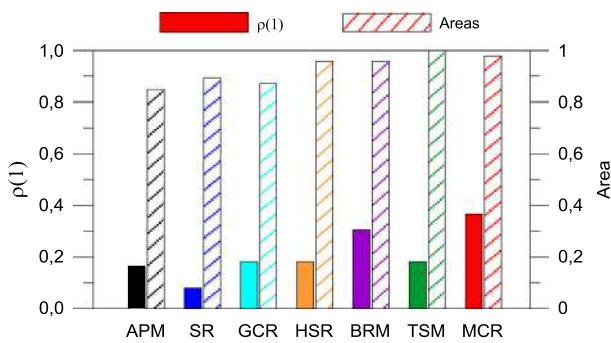


Fig. 3. Global performance of all methods, in all problems.

Fig. 2 presents the PPs corresponding to all experiments in all sets, with the selected metric being the best optimal value. In this figure (and in all the subsequent ones that present PP graphs), the τ axis is split into three sections. The first corresponds to lower values, closer to $\tau = 1$ (thus allowing the assessment of the performance of the methods in terms of providing better optimal solutions). The second section corresponds to intermediate values of τ , while the third presents the final section of the curves, for higher τ values (allowing the assessment of the robustness of the methods in terms of providing more feasible solutions).

Observing the first section of the graph of Fig. 2, one can notice that the curve corresponding to the MCR dominates all other curves. This confirms its better accuracy in terms of the amount of problems for which it provides the best solution, as already verified in Tables 5 and 6. On the other hand, the final sections of the graph indicate that the TSM outperformed the other methods in its ability to provide feasible solutions. This is not surprising, since this method was devised specifically with this purpose, i.e. to prefer feasible solutions over infeasible ones. However, the MCR still shows a very good performance in this respect, with its curve dominated only by that of the TSM method.

A global indication of the performance of all methods may be observed in Fig. 3. This figure contains a bar chart, where the left Y axis and the corresponding solid bars indicate the values of $\rho_s(1)$ that denotes the accuracy of the methods (as explained above). Again, these results indicate the better performance of

the MCR, closely followed by the BRM. Conversely, the right axis and the corresponding hatched bars indicate the areas under the PP curves. Here the TSM method ranks slightly better than the MCR. Again, this reflects the characteristic of the TSM to provide a higher number of feasible solutions, as indicated by the final sections of the $\rho(\tau)$ curves for higher τ values. Still, the MCR performs very well, being followed by the BRM and HSR methods. The other methods do not perform so well, both in terms of accuracy and robustness; the SR provides the smallest number of best optimal solutions (as indicated by its $\rho_s(1)$ value, and its curve for lower τ values in Fig. 2 that confirm the result of Table 5), and is one of the less robust methods.

6. Comparisons for specific sets of experiments

6.1. Set 1: CEC 2006 functions

Fig. 4 presents the PPs corresponding to the experiments with the set of CEC 2006 functions, while Fig. 5 shows the bar charts that summarize the performance measures. The first section of the graph of Fig. 4 (for τ values closer to 1), as well as the solid bars of Fig. 5 representing the ρ_s values for $\tau = 1$, again indicate the better accuracy of the MCR that provides the best optimal solution for a higher number of problems (approximately 37%, as can be seen by the solid red⁴ bar of Fig. 5).

Moreover, considering also the robustness of the methods, the MCR presented the overall best performance for the set of CEC 2006 functions. The performance curve of the MCR dominates the other curves of Fig. 4 for the entire range of τ , except in the final section for higher τ values where the curves for the MCR, TSM and BRM are coincident (it is interesting to observe that, since there is at least one problem for which no feasible solutions were obtained, none of the methods reached $\rho(\tau) = 1$). This overall best performance is confirmed by the areas under the curves represented by the hatched bars of Fig. 5.

The second highest $\rho(1)$ value corresponds to the BRM method, followed by the GCR, again confirming the results of Table 5. However, although those two methods provided a higher number of

⁴ For interpretation of color in Fig. 5, the reader is referred to the web version of this article.

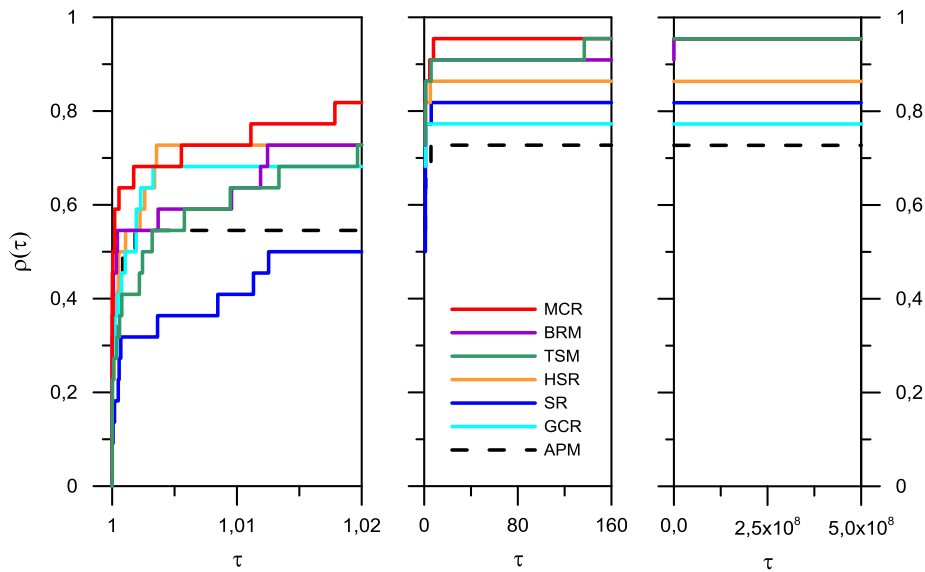


Fig. 4. PPs for the set of CEC 2006 functions.

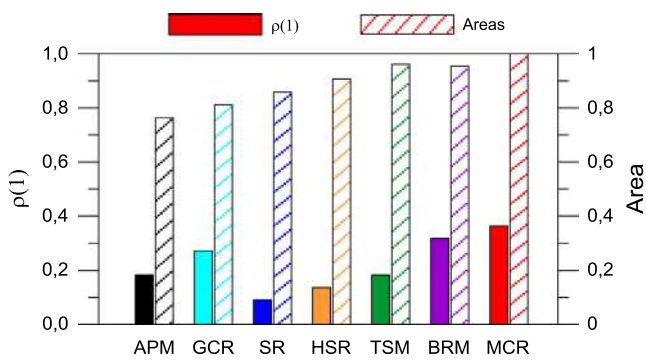


Fig. 5. Global performance for the CEC 2006 functions.

better optimal solutions than the TSM method, this latter method reached the second overall performance as measured by the areas under the curves – this is because its intermediate solutions (that were not the winners for the respective problems) are better than those provided by the BRM and GCR, as indicated by the respective performance curves in the intermediary range of τ values.

Finally, for similar reasons, the HSR and SR methods performed better than the GCR and APM: although these latter methods provided more wins (as indicated in Table 5 and by the $\rho(1)$ values of Fig. 5), their performance curves were dominated by those of the HSR and SR in the intermediary and higher range of τ values.

6.2. Set 2: CEC 2010 functions

Figs. 6 and 7 present the performance measures corresponding to the experiments with the set of CEC 2010 functions. These are complex functions that demand more robustness from the optimization algorithms to provide feasible solutions. In this case,

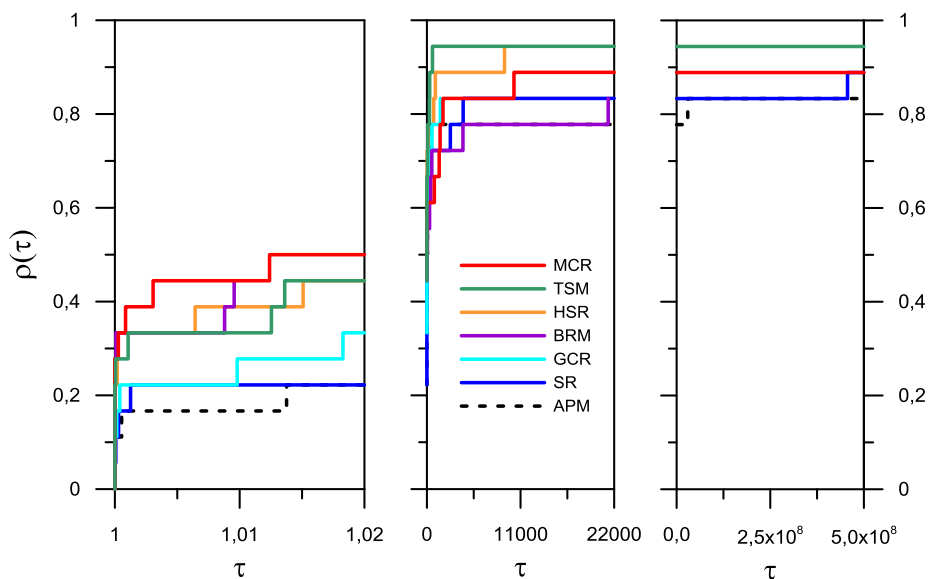


Fig. 6. PPs for the set of CEC 2010 functions.

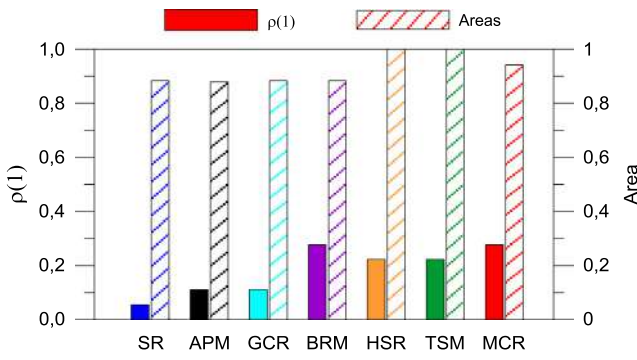


Fig. 7. Global performance for the CEC 2010 functions.

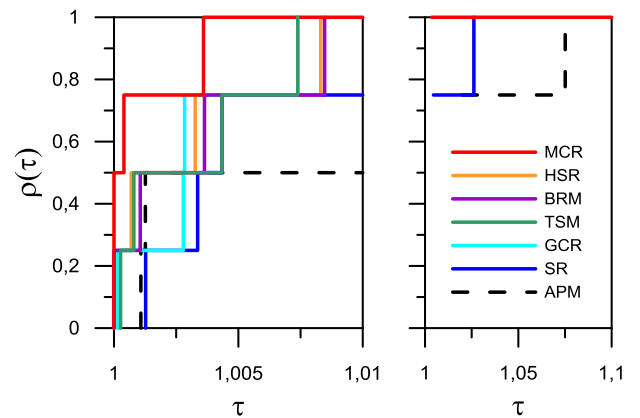


Fig. 10. PPs for the truss problems.

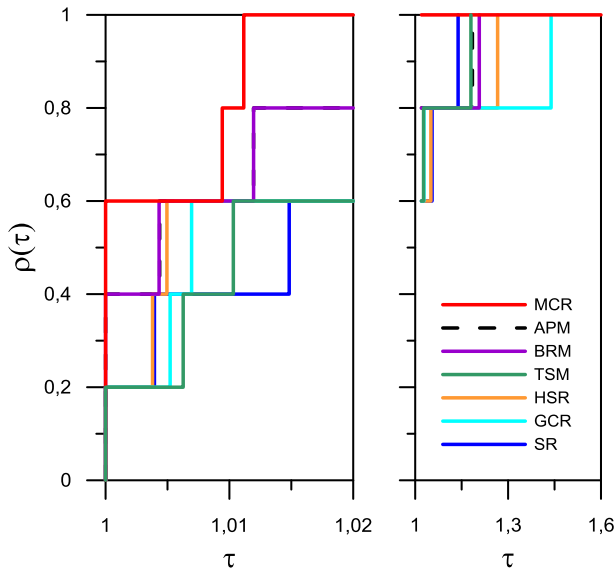


Fig. 8. PPs for the structural engineering problems.

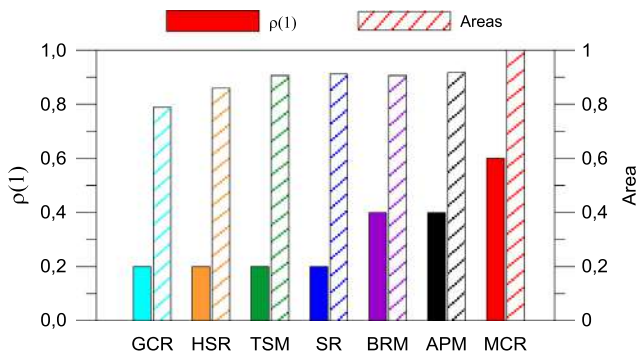


Fig. 9. Global performance for the structural engineering problems.

the TSM method took advantage of its main characteristic (to prefer feasible solutions over infeasible ones), providing a higher number of feasible solutions thus leading to a better robustness; this is indicated by the fact that it was the first to reach the highest $\rho(\tau)$ value, for a smaller value of τ (see the intermediary section of the PP curves in Fig. 6). Because of this, it presented the largest area below the PP curve even though not showing the highest $\rho(1)$ value (only $\approx 16.7\%$, lower than the MCR as seen in the solid bars of Fig. 7). Thus, the TSM can be considered as the best performer for the set of CEC 2010 functions, followed by the HSR and MCR

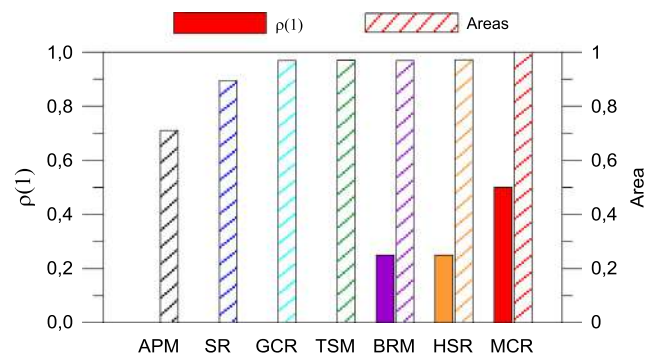


Fig. 11. Global performance for the truss problems.

(even though these latter methods, along with the BRM, could be considered more accurate since they provided better optimal solutions as indicated by their profiles in the lower τ ranges).

6.3. Set 3: structural engineering problems

Fig. 8 presents the PPs corresponding to the experiments with the set of structural engineering problems, while Fig. 9 shows the bar charts that summarize the performance measures. Now, the MCR outperforms the other methods both in terms of accuracy and robustness. It provides the better optimal solutions for 60% of the problems, as can be seen in Fig. 8 for the lower τ values, and also in the solid bars of Fig. 9. Also, it was able to provide feasible solutions for all problems, reaching $\rho(\tau) = 1$ for the lower value of $\tau (\approx 1.01)$.

6.4. Set 4: structural optimization of trusses

Figs. 10 and 11 present the performance measures corresponding to the experiments with the set of truss optimization problems. The MCR is again the best performer, providing the higher number of better solutions, $\rho(1) = 0.5$ as seen in Fig. 10 and in the solid bars of Fig. 11. It also is the most robust, providing $\rho(\tau) = 1$ for the lower value of τ .

6.5. Problems with constraints with different magnitudes

Finally, this sections compares the performance of all methods for the group of problems characterized by constraints with different orders of magnitude and/or different units, as described at the end of Section 4. For this purpose, firstly Table 8 collects the best optimal solutions provided by all methods for this group of prob-

Table 8
Summary of results for the different-magnitude problems.

Functions	MCR	BRM	HSR	SR	GCR	APM	TSM
G10	7093.59	7177.96	7112.12	7333.8	9321.28	–	7188.39
G21	233.224	–	–	–	–	–	–
Pressure Vessel	6064.34	6090.55	6371.86	6394.54	6106.47	6090.83	6102.35
Welded Beam	1.72605	2.08323	2.1854	1.96513	2.48354	2.04497	2.03661
T10C	5062.07	5067.41	5065.58	5079.09	5062.71	5067.54	5063.41
T10D	5534.24	5534.46	5514.4	5538.35	5530.05	5928.68	5538.35
T25D	486.1	490.216	490.14	498.826	489.691	492.117	489.691

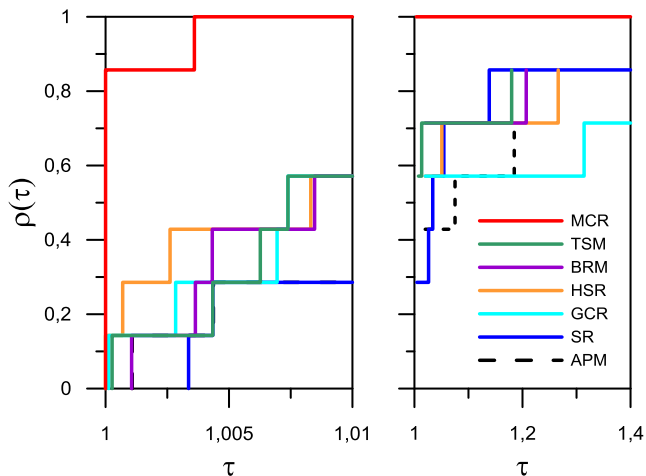


Fig. 12. PPs for the problems with different magnitude constraints.

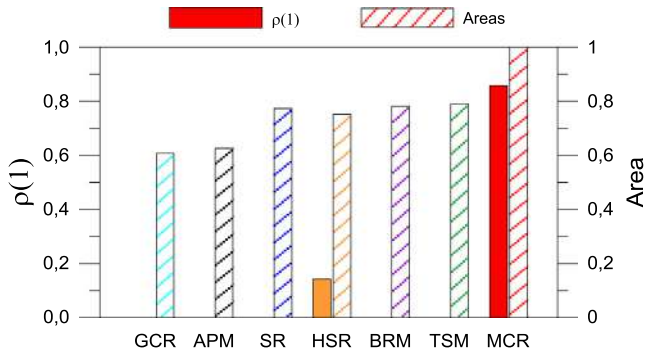


Fig. 13. Global performance for the problems with different magnitude constraints.

lems. Then, Fig. 12 presents the PPs corresponding to the experiments with this group of problems, while Fig. 13 shows the bar charts that summarize the performance measures.

Here the performance of the MCR is outstanding, both in terms of accuracy and robustness. Its performance profile is far above the other methods, for all values of τ . As indicated by the boldface values of Table 8, it provides the best solutions for all problems, with the exception of the T10D truss whose best solution is provided by the HSR. This accuracy of the MCR is reflected by its $\rho(1)$ value of 0.857, as indicated in Figs. 12 and 13; this latter figure confirms that the only other method with a nonzero value for $\rho(1)$ is the HSR.

Considering the CEC 2006 G21 function, the MCR was the only method to provide feasible solutions (for 18 out of 25 runs). This robustness of the MCR is reflected by the hatched bars of Fig. 13, which indicate that the area below its PP curve is noticeably larger than those of the other methods, and also by the fact that it reaches $\rho(\tau) = 1$ for a significantly smaller value of τ as shown in Fig. 12.

7. Conclusions

This work presented a rank-based technique to handle constraints in the solution of optimization problems by evolutionary algorithms. The MCR has been specifically devised for the solution of engineering optimization problems characterized by constraints with different orders of magnitude and/or different units. The MCR was also designed to comprise an “uncoupled” technique that is not embedded into the optimization algorithm, allowing its association with any given evolutionary algorithm; it does not require any user-defined parameter, and is quite simple to implement.

A careful analysis of its performance is presented and compared with six other CHTs, all implemented into the same canonical genetic algorithm, and applied to several well-known benchmark constrained optimization problems. The overall comparisons presented in Section 5 already indicate its better accuracy, not only in terms of the number of wins in the direct comparisons, but mainly in terms of the $\rho_s(1)$ values illustrated in the solid bars of Fig. 3. Although its robustness (measured by the ability of providing more feasible solutions) was slightly outperformed by the TSM, it still presented a very good performance in this respect. The performance of the MCR is significantly enhanced exactly for the class of problems for which it was designed, i.e. those presenting constraints with different magnitudes, as demonstrated in Section 6.5. In summary, the MCR has been shown to be more accurate and robust for those problems, while remaining very competitive for all other sets of problems.

These results indicate the potential of the MCR to efficiently solve practical applications to real-world, complex engineering problems. It is expected that this characteristic may be fully demonstrated with further studies that are presently underway, including the association of the MCR with different evolutionary algorithms, implemented in the computational tool for the optimization of subsea pipeline routes that has been described in [30].

Acknowledgements

The authors acknowledge the support of the Brazilian funding agencies CNPq (grant numbers 306104/2013-0, 305099/2014-0) and FAPERJ (grant numbers E-26/200.314/2016, E-26/201.184/2014, TEC PPM 528/11, TEC PPM 388/14).

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at <http://dx.doi.org/10.1016/j.compstruc.2017.03.023>.

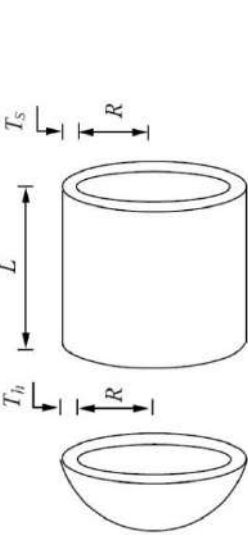
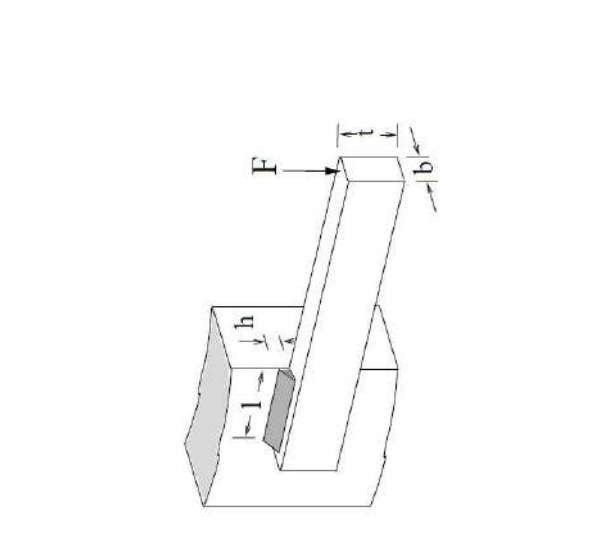
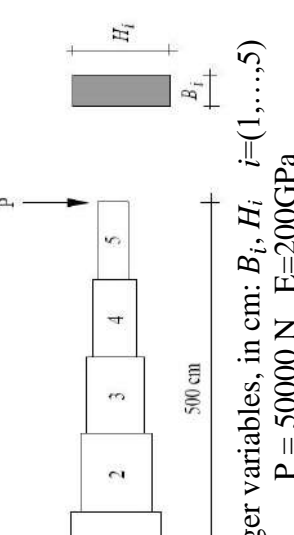
References

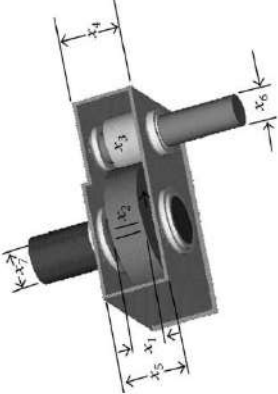
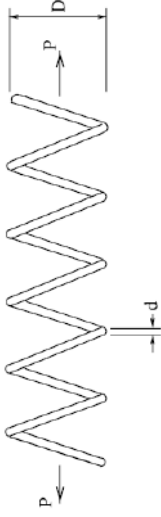
- [1] Shankar N, Hajela P. Heuristics driven strategies for near-optimal structural topology development. In: Topping BHV, editor. Artificial intelligence and structural engineering. Edinburgh, UK: Civil-Comp Press; 1991. <http://dx.doi.org/10.4203/ccp.13.9.1>.

- [2] Lagaros ND, Papadrakakis M, Kokossalakis G. Structural optimization using evolutionary algorithms. *Comput Struct* 2002;80(7–8):571–89.
- [3] Schmid H, Thierauf G. A combined heuristic optimization technique. *Adv Eng Softw* 2005;36:11–9.
- [4] Saka MP. Optimum design of steel frames using stochastic search techniques based on natural phenomena: a review. In: Topping BHV, editor. *Civil engineering computations: tools and techniques*. Stirlingshire, UK: Saxe-Coburg Publications; 2007. <http://dx.doi.org/10.4203/csets.16.6> [chapter 6].
- [5] Hasańcebi O, Çarbas S, Dogan E, Erdal F, Saka MP. Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. *Comput Struct* 2009;87:284–302.
- [6] Liu D, Toropov VV. A lamination parameter-based strategy for solving an integer-continuous problem arising in composite optimization. *Comput Struct* 2013;128:170–4.
- [7] Sahab MG, Toropov VV, Gandomi AH. A review on traditional and modern structural optimization: problems and techniques. In: Gandomi AH, Yang XS, Talatahari S, Alavi AH, editors. *Metaheuristic applications in structures and infrastructures*. Oxford: Elsevier; 2013. p. 25–47 [ISBN: 9780123983640].
- [8] Hare W, Nutini J, Tesfamariam S. A survey of non-gradient optimization methods in structural engineering. *Adv Eng Softw* 2013;59:19–28.
- [9] Vieira IN, de Lima BSLP, Jacob BP. Bio-inspired algorithms for the optimization of offshore oil production systems. *Int J Numer Meth Eng* 2012;91:1023–44. <http://dx.doi.org/10.1002/nme.4301>.
- [10] Holland JH. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press; 1975.
- [11] Goldberg DE. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley; 1989.
- [12] Kennedy J, Eberhardt RC. Particle swarm optimization. In: *Proceedings IEEE conference on neural networks*. p. 1942–8.
- [13] Kennedy J, Eberhardt RC, Shi Y. *Swarm intelligence*. The Morgan Kaufmann series in evolutionary computation. San Francisco: Academic Press; 2001.
- [14] Dasgupta D. *Artificial immune systems and their applications*. Springer-Verlag; 1998.
- [15] de Castro LN, von Zuben FJV. Learning and optimization using the clonal selection principle. In: *IEEE Trans Evol Comput Spec Iss Artif Immune Syst*.
- [16] Dorigo M, Stützle T. *Ant colony optimization*. MIT Press; 2004.
- [17] Askarzadeh A. A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Comput Struct* 2016;169:1–12.
- [18] Bäck T. *Evolutionary algorithms in theory and practice*. New York: Oxford University Press; 1996.
- [19] Engelbrecht AP. *Fundamentals of computational swarm intelligence*. John Wiley & Sons; 2005.
- [20] Mezura-Montes E, Coello Coello CA. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and evolutionary computation*, vol. 1. Elsevier; 2011. p. 173–94.
- [21] Michalewicz Z, Schoenauer M. Evolutionary algorithms for constrained parameter optimization problems. *Evol Comput* 1996;4:1–32.
- [22] Coello Coello CA. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Comput Methods Appl Mech Eng* 2002;191:1245–87.
- [23] Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization. *IEEE Trans Evol Comput* 2000;4:284–94.
- [24] Runarsson TP, Yao X. Continuous selection and self-adaptive evolution strategies. *Proceedings of the 2002 congress on evolutionary computation – CEC'02*, vol. 1. p. 279–84.
- [25] Ho PY, Shimizu K. Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. *Inf Sci* 2007;177:2985–3004.
- [26] Rodrigues MC, de Lima BSLP, Guimarães S. Balanced ranking method for constrained optimization problems using evolutionary algorithms. *Inf Sci* 2016;327:71–90. <http://dx.doi.org/10.1016/j.ins.2015.08.012>.
- [27] de Lima BSLP, Jacob BP, Ebecken NFF. A hybrid fuzzy/genetic algorithm for the design of offshore oil production risers. *Int J Numer Meth Eng* 2005;64:1459–82. <http://dx.doi.org/10.1002/nme.1416>.
- [28] de Pina AA, Albrecht CH, de Lima BSLP, Jacob BP. Tailoring the particle swarm optimization algorithm for the design of offshore oil production risers. *Optim Eng* 2011;12:215–35. <http://dx.doi.org/10.1007/s11081-009-9103-5>.
- [29] de Lucena RR, de Lima BSLP, Jacob BP, Rocha DM. Optimization of pipeline routes using an AIS/adaptive penalty method. In: Topping BHV, editor. *Proceedings of the eighth international conference on engineering computational technology*. Stirlingshire, UK: Civil-Comp Press; 2012. <http://dx.doi.org/10.4203/ccp.100.66> [paper 66].
- [30] de Lucena RR, Baioco JS, de Lima BSLP, Albrecht CH, Jacob BP. Optimal design of submarine pipeline routes by genetic algorithm with different constraint handling techniques. *Adv Eng Softw* 2014;76:110–24. <http://dx.doi.org/10.1016/j.advengsoft.2014.06.003>.
- [31] Det Norske Veritas. On bottom stability design of submarine pipelines, recommended practice DNV-RP-F109; 2008.
- [32] Det Norske Veritas. Free spanning pipelines, recommended practice DNV-RP-F105; February 2006.
- [33] Mallipeddi R, Suganthan PN. Ensemble of constraint handling techniques. *IEEE Trans Evol Comput* 2010;14:561–79.
- [34] Barbosa HJC, Lemonge ACC. A new adaptive penalty scheme for genetic algorithms. *Inf Sci* 2003;156:215–51.
- [35] Lemonge ACC, Barbosa HJC. An adaptive penalty scheme for genetic algorithms in structural optimization. *Int J Numer Meth Eng* 2004;59:703–36. <http://dx.doi.org/10.1002/nme.899>.
- [36] Deb K. An efficient constraint handling method for genetic algorithms. *Comput Methods Appl Mech Eng* 2000;186(2–4):311–38.
- [37] Liepins GE. A genetic algorithm approach to multiple-fault diagnosis. In: *Handbook of genetic algorithms*.
- [38] Orvosh D, Lawrence D. Using a genetic algorithm to optimize problems with feasibility constraints. In: *IEEE world congress on computational intelligence*.
- [39] Coello Coello CA. Use of a self-adaptive penalty approach for engineering optimization problems. *Comput Ind* 2000;41:113–27.
- [40] Lin C-Y, Wu W-H. Self-organizing adaptive penalty strategy in constrained genetic search. *Struct Multidisc Optim* 2004;26:417–28.
- [41] Wu W-H, Lin C-Y. The second generation of self-organizing adaptive penalty strategy for constrained genetic search. *Adv Eng Softw* 2004;35:815–25.
- [42] Garcia RP, de Lima BSLP, Jacob BP, Lemonge ACC. Handling optimization problems with constraints of different magnitudes using evolutionary algorithms. In: *CILAMCE 2015 - XXXVI Iberian Latin-American congress on computational methods in engineering*. Rio de Janeiro – RJ.
- [43] Eshelman LJ, Schaffer JD. Crossover's niche. In: *Proceedings of the 5th international conference on genetic algorithms*. Morgan Kaufmann Publishers Inc.; 1993.
- [44] Liang JJ, Runarsson TP, Mezura-Montes E, Clerc M, Suganthan PN, Coello Coello CA, et al. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization [technical report]. Singapore: School of EEE, Nanyang Technological University; 2006.
- [45] Mallipeddi R, Suganthan PN. Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. Singapore: Nanyang Technological University; 2010.
- [46] Sandgren E. Nonlinear integer and discrete programming in mechanical design. In: *Proceedings of the ASME design technology conference*, Kissimmee, FL. p. 95–105.
- [47] Deb K. Optimal design of a welded beam via genetic algorithms. *AIAA J* 1991;29(11):2012–5.
- [48] Erbaturo F, Hasańcebi O, Tütüncü İ, Kılıç H. Optimal design of planar and space structures with genetic algorithms. *Comput Struct* 2000;75:209–24.
- [49] Gellatly RA, Berke L. Optimal structural design [Technical report no. AFFDL-TR-70-165]. Fairborn, OH: Air Force Flight Dynamics Lab., AFFDL; 1971.
- [50] Krishnamoorthy CS, Rajeev S. Discrete optimization of structures using genetic algorithms. *J Struct Eng* 1992;118(5):1233–50.
- [51] Wu SJ, Chow PT. Steady-state genetic algorithms for discrete optimization of trusses. *Comput Struct* 1995;56(6):979–91.
- [52] Sheskin DJ. *Handbook of parametric and nonparametric statistical procedures*, vol. 1736. London/West Palm Beach: Chapman & Hall/CRC; 2006.
- [53] 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 Evol Comput* 2011;1:3–18.
- [54] Dolan ED, Moré JJ. Benchmarking optimization software with performance profiles. *Math Program Ser A* 2002;91:201–13. <http://dx.doi.org/10.1007/s101070100263>.
- [55] Gibbons JD, Chakraborti S. *Nonparametric statistical inference*. 5th ed. Marcel Dekker Inc.; 2003.
- [56] Barreto AM, Bernardino HS, Barbosa HJ. Probabilistic performance profiles for the experimental evaluation of stochastic algorithms. In: *Proceedings of the 12th annual conference on genetic and evolutionary computation, GECCO'10*. Portland, Oregon: ACM; 2010. p. 751–8.
- [57] Costa L, Espirito Santo I, Oliveira P. Stochastic algorithms assessment using performance profiles. In: *Proceedings of the 13th annual conference on genetic and evolutionary computation, GECCO'11*. Dublin, Ireland: ACM; 2011.

Appendix A – Benchmark Problems

Structural Engineering Problems

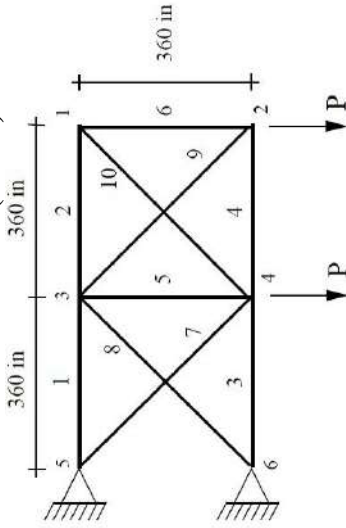
Name	Variables	Function	Constraints	Bounds
<p style="text-align: center;">Pressure Vessel</p>		$W(T_s, T_h, R, L) = 0.6224T_sLR + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2R$	$g_1(T_s, R) = 0.0193R - T_s \leq 0$ $g_2(T_h, R) = 0.00954R - T_h \leq 0$ $g_3(R, L) = 1296000 - \pi R^2L - \frac{4}{3}\pi R^3 \leq 0$ $g_4(L) = L - 240 \leq 0$	<p>0.0625 ≤ T_s, T_h ≤ 5 (discrete variables in constant steps of 0.0625) 10 ≤ R, L ≤ 200</p>
<p style="text-align: center;">Welded Beam</p>		$C(h, l, t, b) = 1.10471h^2l + 0.04811tb(14.0 + l)$	<p>Shear stress, bending stress, buckling load, end deflection, side constraints.</p> $g_1(\tau) = \tau - 13600 \leq 0$ $g_2(\sigma) = \sigma - 30000 \leq 0$ $g_3(b, h) = h - b \leq 0$ $g_4(h, t, b, l) = 0.10471h^2 + 0.04811tb(14.0 + l) - 5.0 \leq 0$ $g_5(h) = 0.125 - h \leq 0$ $g_6(\delta) = \delta - 0.25 \leq 0$ $g_7(P_c) = 6000 - P_c \leq 0$ <p>Where:</p> $\tau = \sqrt{(\tau')^2 + (\tau''^2 + l\tau'\tau''/\alpha)} \quad \tau' = \frac{6000}{\sqrt{2}ht}$ $\alpha = \sqrt{0.25(l^2 + (h+t)^2)}$ $\sigma = \frac{504000}{t^2b}$ $P_c = 64746.022(1 - 0.0282346t)tb^3$ $\delta = \frac{2.1952}{t^3b} \quad \tau'' = \frac{6000(14+0.5l)\alpha}{2(0.707hl(l^2/12 + 0.25(h+t)^2))}$	<p>0.125 ≤ h ≤ 10 0.1 ≤ l, t, b ≤ 10</p>
<p style="text-align: center;">Cantilever Beam</p>	 <p>Integer variables, in cm: $B_i, H_i \quad i=(1, \dots, 5)$ $P = 50000 \text{ N} \quad E=200\text{GPa}$</p>	$V(H_i, B_i) = 100 \sum_{i=1}^5 H_i B_i$	<p>Stress σ_i, Deflection δ</p> $g_i(H_i, B_i) = \sigma_i - \frac{14000N}{\text{cm}^2} \leq 0 \quad i = 1, \dots, 5$ $g_{i+5}(H_i, B_i) = \frac{H_i}{B_i} - 20 \leq 0 \quad i = 1, \dots, 5$ $g_{11}(H_i, B_i) = \delta - 2.7\text{cm} \leq 0$	<p>$B_2, B_3 \in \{2.4, 2.6, 2.8, 3.1\}$, $H_2, H_3 \in \{45.0, 50.0, 55.0, 60.0\}$ B_4, H_4, B_5 and H_5 are continuous.</p>

Name	Variables	Function	Constraints	Bounds
<p>Speed Reducer</p>	 <p>face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, diameter of the second shaft x_7. All variables continuous except x_3.</p>	$f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$	<p>Bending stress of the gear teeth, surface stress, transverse deflections of shafts and stresses in shaft</p> $g_1(\vec{x}) = \frac{x_1x_2^2x_3}{27} - 1 \leq 0$ $g_2(\vec{x}) = \frac{x_1x_2^2x_3}{397.5} - 1 \leq 0$ $g_3(\vec{x}) = \frac{x_2x_3x_4^4}{1.93x_3^4} - 1 \leq 0$ $g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0$ $g_5(\vec{x}) = \frac{1.0}{110x_6^3} \sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} - 1 \leq 0$ $g_6(\vec{x}) = \frac{1.0}{85x_7^3} \sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} - 1 \leq 0$ $g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \leq 0 \quad g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \leq 0$ $g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \leq 0 \quad g_{10}(\vec{x}) = \frac{1.5x_6+1.9}{x_4} - 1 \leq 0$ $g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$	$2.6 \leq x_1 \leq 3.6$ $0.7 \leq x_2 \leq 0.8$ $17 \leq x_3 \leq 28$ $7.3 \leq x_4 \leq 8.3$ $7.8 \leq x_5 \leq 8.3$ $2.9 \leq x_6 \leq 3.9$ $5.0 \leq x_7 \leq 5.5$
<p>Tension/Compression Spring</p>	 <p>Mean coil diameter D, wire diameter d, number of active coils N.</p>	$f(\vec{x}) = (N + 2)Dd^2$	<p>Minimum deflection, shear stress, surge frequency, outside diameter</p> $g_1(\vec{x}) = 1 - \frac{D^3N}{71785d^4} \leq 0$ $g_2(\vec{x}) = \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{5108d^2}{140.45d} - 1 \leq 0$ $g_3(\vec{x}) = 1 - \frac{D^2N}{140.45d} \leq 0$ $g_4(\vec{x}) = \frac{D + d}{1.5} - 1 \leq 0$	$0.05 \leq d \leq 2$ $0.25 \leq D \leq 1.3$ $2 \leq N \leq 1.5$

Structural Optimization of Trusses

Variables	Function	Constraints	Bounds
A_i (cross-section area of the bars). ρ_i and l_i : density and length of i_{th} bar; e : total number of bars.	$weight(A_i) = \sum_{i=1}^e \rho_i l_i A_i$	$g_1(\delta_i) = \delta_i - \delta_{max} \leq 0, \quad i = 1, \dots, n$ $g_2(\sigma_i) = \sigma_i - \sigma_{max} \leq 0, \quad i = 1, \dots, e$ n : number of nodes; δ_i : nodal displacements σ_i : stresses.	$A_i \in S$

10-bar truss: continuous (T10C) and discrete (T10D) cases



Material density = 0.1 lb/in^3 , Young's modulus = 10^4 ksi

Vertical downward loads $P = 100 \text{ kips}$

Allowable stresses: $\pm 25 \text{ ksi}$

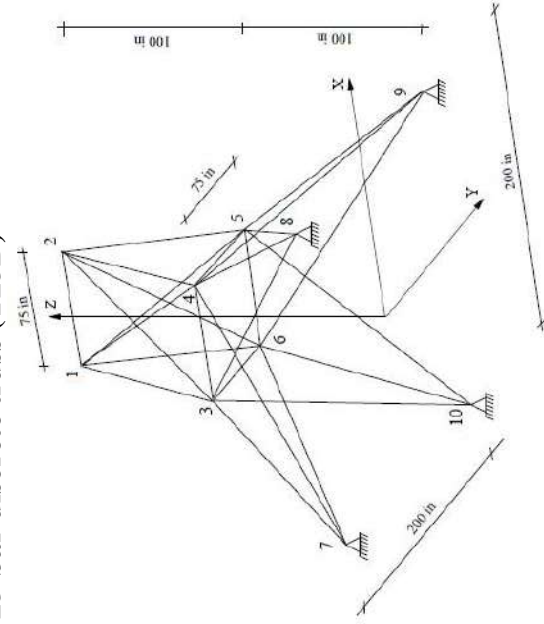
Max. displacements: 2 in both horizontal vertical directions.

Discrete case: Cross-section areas chosen from

- $S = \{1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.93, 3.13, 3.38, 3.47, 3.55, 3.63, 3.88, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.97, 11.50, 13.50, 14.20, 15.50, 16.90, 18.80, 19.90, 22.00, 26.50, 30.00, 33.50\} \text{ (in}^2\text{)}$

Continuous case: $S = [0.1 \text{ to } 33.50]$.

25-bar discrete truss (T25D)



Member Group	Connectivity
A_1	1-2
A_2	1-4, 2-3, 1-5, 2-6
A_3	2-5, 2-4, 1-3, 1-6
A_4	3-6, 4-5
A_5	3-4, 5-6
A_6	3-10, 6-7, 4-9, 5-8
A_7	3-8, 4-7, 6-9, 5-10
A_8	3-7, 4-8, 5-9, 6-10

Node	Loads	
	F_x	F_y
1	1	-10
2	0	-10
3	0.5	0
6	0.6	0

Allowable stresses: $\pm 40 \text{ ksi}$;

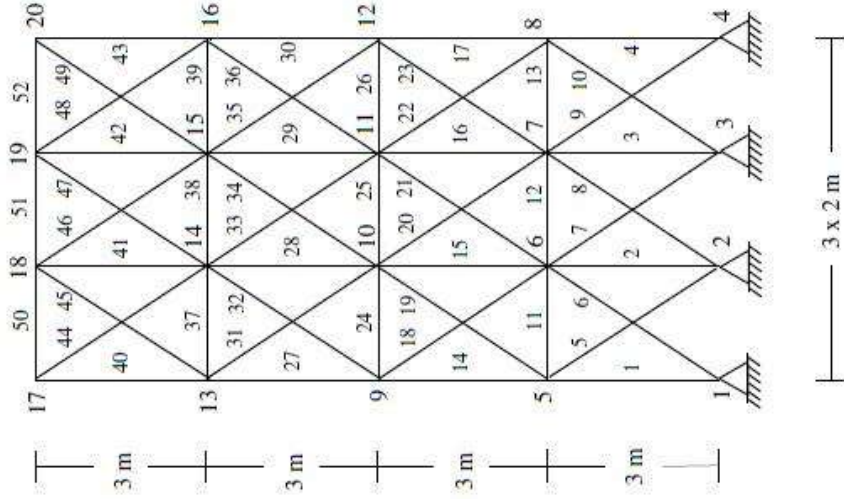
Max. displacements at nodes 1 and

2 (x and y directions): 0.35 in .

- $S = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.1, 3.2\} \text{ (in}^2\text{)}$.

Material density = 0.1 lb/in^3 , Young's modulus = 10^4 ksi

52-bar truss discrete (T52D)



Member Group	Connectivity
A ₁	1, 2, 3, 4
A ₂	5, 6, 7, 8, 9, 10
A ₃	11, 12, 13
A ₄	14, 15, 16, 17
A ₅	18, 19, 20, 21, 22, 23
A ₆	24, 25, 26
A ₇	27, 28, 29, 30
A ₈	31, 32, 33, 34, 35, 36
A ₉	37, 38, 39
A ₁₀	40, 41, 42, 43
A ₁₁	44, 45, 46, 47, 48, 49
A ₁₂	50, 51, 52

Node	Loads	
	F _x	F _y
17	100	200
18	100	200
19	100	200
20	100	200

Cross-section areas (in mm²) chosen from $S = \{71.613, 90.968, 126.451, 161.290, 198.064, 252.258, 285.161, 363.225, 388.386, 494.193, 506.451, 641.289, 645.160, 792.256, 816.773, 940.000, 1008.385, 1045.159, 1161.288, 1283.868, 1374.191, 1535.481, 1690.319, 1696.771, 1858.061, 1890.319, 1993.544, 2019.351, 2180.641, 2238.705, 2290.318, 2341.191, 2477.414, 2496.769, 2503.221, 2696.769, 2722.575, 2896.768, 2961.284, 3096.768, 3206.445, 3303.219, 3703.218, 4658.055, 5141.925, 5503.215, 5999.998, 6999.986, 7419.340, 8709.660, 8967.724, 9161.272, 9999.980, 10322.560, 10903.204, 12129.008, 12838.684, 14193.520, 14774.164, 15806.420, 17096.740, 18064.480, 19354.800, 21612.860\}$.

Allowable stress in tension and compression are both 180 MPa. There is no displacement constraint.

Material density = 7860 kg/m³,
Young's modulus = 2.07 × 10⁵ MPa

CEC 2006 Functions

Name	Function	Constraints	Bounds
G01	$f(\vec{x}) = 5 \sum_{i=1}^4 x_i - 5 \sum_{i=1}^4 x_i^2 - \sum_{i=5}^{13} x_i$	$g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$ $g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$ $g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$ $g_4(\vec{x}) = -8x_1 + x_{10} \leq 0$ $g_5(\vec{x}) = -8x_2 + x_{11} \leq 0$ $g_6(\vec{x}) = -8x_3 + x_{12} \leq 0$ $g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leq 0$ $g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leq 0$ $g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leq 0$	$0 \leq x_i \leq 1 \ (i = 1, \dots, 9)$ $0 \leq x_i \leq 100 \ (i = 10, \dots, 12)$ $0 \leq x_{13} \leq 1$
G02	$f(\vec{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 $	$g_1(\vec{x}) = 0.75 - \prod_{i=1}^n x_i \leq 0$ $g_2(\vec{x}) = \sum_{i=1}^n x_i - 7.5n \leq 0$	<p>Where $n = 20$</p> $0 \leq x_i \leq 10 \ (i = 1, \dots, n)$
G03	$f(\vec{x}) = -(\sqrt{n})^n \prod_{i=1}^n x_i$	$h_1(\vec{x}) = \sum_{i=1}^n x_i^2 - 1 = 0$	<p>Where $n = 10$</p> $0 \leq x_i \leq 1 \ (i = 1, \dots, n)$
G04	$f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$	$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$ $g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$ $g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$ $g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$ $g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$ $g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$	$78 \leq x_1 \leq 102$ $33 \leq x_2 \leq 45$ $27 \leq x_i \leq 45 \ (i = 3, 4, 5)$
G05	$f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + \left(\frac{0.000002}{3} \right) x_2^3$	$g_1(\vec{x}) = -x_4 + x_3 - 0.55 \leq 0$ $g_2(\vec{x}) = -x_3 + x_4 - 0.55 \leq 0$ $h_3(\vec{x}) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0$ $h_4(\vec{x}) = 1000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$ $h_5(\vec{x}) = 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0$	$0 \leq x_1 \leq 1200$ $0 \leq x_2 \leq 1200$ $-0.55 \leq x_3 \leq 0.55$ $-0.55 \leq x_4 \leq 0.55$
G06	$f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$	$g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$ $g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$	$13 \leq x_1 \leq 100$ $0 \leq x_2 \leq 100$

Name	Function	Constraints	Bounds
G07	$f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$	$g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$ $g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$ $g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$ $g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$ $g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$ $g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$ $g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$ $g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$	$-10 \leq x_i \leq 10 \quad (i = 1, \dots, 10)$
G08	$f(\vec{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$	$g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0$ $g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$	$0 \leq x_1 \leq 10$ $0 \leq x_2 \leq 10$
G09	$f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$	$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$ $g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$ $g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$ $g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$	$-10 \leq x_i \leq 10 \quad (i = 1, \dots, 7)$
G10	$f(\vec{x}) = x_1 + x_2 + x_3$	$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$ $g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$ $g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$ $g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$ $g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$ $g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$	$100 \leq x_1 \leq 10000$ $1000 \leq x_i \leq 10000 \quad (i = 2, 3)$ $10 \leq x_i \leq 1000 \quad (i = 4, \dots, 8)$
G11	$f(\vec{x}) = x_1^2 + (x_2 - 1)^2$	$h_1(\vec{x}) = x_2 - x_1^2 = 0$	$-1 \leq x_1 \leq 1$ $-1 \leq x_2 \leq 1$
G12	$f(\vec{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}$	$g_1(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$	$0 \leq x_i \leq 10 \quad (i = 1, 2, 3)$ and $p, q, r = 1, 2, \dots, 9$
G13	$f(\vec{x}) = e^{x_1x_2x_3x_4x_5}$	$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$ $h_2(\vec{x}) = x_2x_3 - 5x_4x_5 = 0$ $h_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$	$-2.3 \leq x_i \leq 2.3 \quad (i = 1, 2)$ $-3.2 \leq x_i \leq 3.2 \quad (i = 3, 4, 5)$
G14	$f(\vec{x}) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$	$h_1(\vec{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$ $h_2(\vec{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$ $h_3(\vec{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$	$0 \leq x_i \leq 10 \quad (i = 1, \dots, 10)$ and $c_1 = -6.089, c_2 = -17.164,$ $c_3 = -34.054, c_4 = -5.914,$ $c_5 = -24.721, c_6 = -14.986,$ $c_7 = -24.1, c_8 = -10.708, c_9 = -26.662, c_{10} = -22.179$
G15	$f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$	$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$ $h_2(\vec{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$	$0 \leq x_i \leq 10 \quad (i = 1, 2, 3)$

Name	Function	Constraints	Bounds
G16	$ \begin{aligned} f(\vec{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} \end{aligned} $	$ \begin{aligned} g_1(\vec{x}) &= \frac{0.28}{0.72}y_5 - y_4 \leq 0 & g_2(\vec{x}) &= x_3 - 1.5x_2 \leq 0 \\ g_3(\vec{x}) &= 3496 \frac{y_2}{c_{12}} - 21 \leq 0 & g_4(\vec{x}) &= 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0 \\ g_5(\vec{x}) &= 213.1 - y_1 \leq 0 & g_6(\vec{x}) &= y_1 - 405.23 \leq 0 \\ g_7(\vec{x}) &= 17.505 - y_2 \leq 0 & g_8(\vec{x}) &= y_2 - 1053.6667 \leq 0 \\ g_9(\vec{x}) &= 11.275 - y_3 \leq 0 & g_{10}(\vec{x}) &= y_3 - 35.03 \leq 0 \\ g_{11}(\vec{x}) &= 214.228 - y_4 \leq 0 & g_{12}(\vec{x}) &= y_4 - 665.585 \leq 0 \\ g_{13}(\vec{x}) &= 7.458 - y_5 \leq 0 & g_{14}(\vec{x}) &= y_5 - 584.463 \leq 0 \\ g_{15}(\vec{x}) &= 0.961 - y_6 \leq 0 & g_{16}(\vec{x}) &= y_6 - 265.916 \leq 0 \\ g_{17}(\vec{x}) &= 1.612 - y_7 \leq 0 & g_{18}(\vec{x}) &= y_7 - 7.046 \leq 0 \\ g_{19}(\vec{x}) &= 0.146 - y_8 \leq 0 & g_{20}(\vec{x}) &= y_8 - 0.222 \leq 0 \\ g_{21}(\vec{x}) &= 107.99 - y_9 \leq 0 & g_{22}(\vec{x}) &= y_9 - 273.366 \leq 0 \\ g_{23}(\vec{x}) &= 922.693 - y_{10} \leq 0 & g_{24}(\vec{x}) &= y_{10} - 1286.105 \leq 0 \\ g_{25}(\vec{x}) &= 926.832 - y_{11} \leq 0 & g_{26}(\vec{x}) &= y_{11} - 1444.046 \leq 0 \\ g_{27}(\vec{x}) &= 18.766 - y_{12} \leq 0 & g_{28}(\vec{x}) &= y_{12} - 537.141 \leq 0 \\ g_{29}(\vec{x}) &= 1072.163 - y_{13} \leq 0 & g_{30}(\vec{x}) &= y_{13} - 3247.039 \leq 0 \\ g_{31}(\vec{x}) &= 8961.448 - y_{14} \leq 0 & g_{32}(\vec{x}) &= y_{14} - 26844.086 \leq 0 \\ g_{33}(\vec{x}) &= 0.063 - y_{15} \leq 0 & g_{34}(\vec{x}) &= y_{15} - 0.386 \leq 0 \\ g_{35}(\vec{x}) &= 71084.33 - y_{16} \leq 0 & g_{36}(\vec{x}) &= -140000 + y_{16} \leq 0 \\ g_{37}(\vec{x}) &= 2802713 - y_{17} \leq 0 & g_{38}(\vec{x}) &= y_{17} - 12146108 \leq 0 \end{aligned} $ <p>Where:</p> $ \begin{aligned} 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 + 0.580y_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 \end{aligned} $	$ \begin{aligned} 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 \end{aligned} $

Name	Function	Constraints	Bounds																																																																																																												
G17	$f(\vec{x}) = f(x_1) + f(x_2)$ $f(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \\ 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$	$h_1(\vec{x}) = -x_1 + 300 - \frac{x_3x_4}{131.078} \cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078} \cos(1.47588) = 0$ $h_2(\vec{x}) = -x_2 - \frac{x_3x_4}{131.078} \cos(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078} \cos(1.47588) = 0$ $h_3(\vec{x}) = -x_5 - \frac{x_3x_4}{131.078} \sin(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078} \sin(1.47588) = 0$ $h_4(\vec{x}) = 200 - \frac{x_3x_4}{131.078} \sin(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078} \sin(1.47588) = 0$	$0 \leq x_1 \leq 400$ $0 \leq x_2 \leq 1000$ $340 \leq x_3 \leq 420$ $340 \leq x_4 \leq 420$ $-1000 \leq x_5 \leq 1000$ $0 \leq x_6 \leq 0.5236$																																																																																																												
G18	$f(\vec{x}) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$	$g_1(\vec{x}) = x_3^2 + x_4^2 - 1 \leq 0 \quad g_2(\vec{x}) = x_6^2 - 1 \leq 0$ $g_3(\vec{x}) = x_5^2 + x_6^2 - 1 \leq 0 \quad g_4(\vec{x}) = x_1^2 + (x_2 - x_9)^2 - 1 \leq 0$ $g_5(\vec{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0$ $g_6(\vec{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$ $g_7(\vec{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0$ $g_8(\vec{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0$ $g_9(\vec{x}) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0 \quad g_{10}(\vec{x}) = x_2x_3 - x_1x_4 \leq 0$ $g_{11}(\vec{x}) = -x_3x_9 \leq 0 \quad g_{12}(\vec{x}) = x_5x_9 \leq 0$ $g_{13}(\vec{x}) = x_6x_7 - x_5x_8 \leq 0$	$-10 \leq x_i \leq 10 \quad (i = 1, \dots, 8)$ $0 \leq x_9 \leq 20$																																																																																																												
G19	$f(\vec{x}) = \sum_{j=1}^5 \sum_{i=1}^5 c_{ij} x_{(10+i)} x_{(10+j)} + 2 \sum_{j=1}^5 d_j x_{(10+j)}^3$ $\vec{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$	$g_j(\vec{x}) = -2 \sum_{i=1}^5 c_{ij} x_{(10+i)} - 3d_j x_{(10+j)}^2 - e_j + \sum_{i=1}^5 a_{ij} x_i \leq 0 \quad j = 1, \dots, 5$ <table border="1" data-bbox="813 716 1468 1232"> <thead> <tr> <th>j</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> </tr> </thead> <tbody> <tr> <td>e_j</td> <td>-15</td> <td>-27</td> <td>-36</td> <td>-18</td> <td>-12</td> </tr> <tr> <td>c_{1j}</td> <td>30</td> <td>-20</td> <td>-10</td> <td>32</td> <td>-10</td> </tr> <tr> <td>c_{2j}</td> <td>-20</td> <td>39</td> <td>-6</td> <td>-31</td> <td>32</td> </tr> <tr> <td>c_{3j}</td> <td>-10</td> <td>-6</td> <td>10</td> <td>-6</td> <td>-10</td> </tr> <tr> <td>c_{4j}</td> <td>32</td> <td>-31</td> <td>-6</td> <td>39</td> <td>-20</td> </tr> <tr> <td>c_{5j}</td> <td>-10</td> <td>32</td> <td>-10</td> <td>-20</td> <td>30</td> </tr> <tr> <td>d_j</td> <td>4</td> <td>8</td> <td>10</td> <td>6</td> <td>2</td> </tr> <tr> <td>a_{1j}</td> <td>-16</td> <td>2</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>a_{2j}</td> <td>0</td> <td>-2</td> <td>0</td> <td>0.4</td> <td>2</td> </tr> <tr> <td>a_{3j}</td> <td>-3.5</td> <td>0</td> <td>2</td> <td>0</td> <td>0</td> </tr> <tr> <td>a_{4j}</td> <td>0</td> <td>-2</td> <td>0</td> <td>-4</td> <td>-1</td> </tr> <tr> <td>a_{5j}</td> <td>0</td> <td>-9</td> <td>-2</td> <td>1</td> <td>-2.8</td> </tr> <tr> <td>a_{6j}</td> <td>2</td> <td>0</td> <td>-4</td> <td>0</td> <td>0</td> </tr> <tr> <td>a_{7j}</td> <td>-1</td> <td>-1</td> <td>-1</td> <td>-1</td> <td>-1</td> </tr> <tr> <td>a_{8j}</td> <td>-1</td> <td>-2</td> <td>-3</td> <td>-2</td> <td>-1</td> </tr> <tr> <td>a_{9j}</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> <tr> <td>a_{10j}</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	j	1	2	3	4	5	e_j	-15	-27	-36	-18	-12	c_{1j}	30	-20	-10	32	-10	c_{2j}	-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	$0 \leq x_i \leq 10 \quad (i = 1, \dots, 15)$
j	1	2	3	4	5																																																																																																										
e_j	-15	-27	-36	-18	-12																																																																																																										
c_{1j}	30	-20	-10	32	-10																																																																																																										
c_{2j}	-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																																																																																																										

Name	Function	Constraints	Bounds
G21	$f(\vec{x}) = x_1$	$g_1(\vec{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$ $h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$ $h_2(\vec{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$ $h_3(\vec{x}) = -x_5 + \ln(-x_4 + 900) = 0$ $h_4(\vec{x}) = -x_6 + \ln(x_4 + 300) = 0$ $h_5(\vec{x}) = -x_7 + \ln(-2x_4 + 700) = 0$	$0 \leq x_1 \leq 1000$ $0 \leq x_2, x_3 \leq 40$ $100 \leq x_4 \leq 300$ $6.3 \leq x_5 \leq 6.7$ $5.9 \leq x_6 \leq 6.4$ $4.5 \leq x_7 \leq 6.25$
G23	$f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$	$g_1(\vec{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$ $g_2(\vec{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$ $h_1(\vec{x}) = x_1 + x_2 - x_3 - x_4 = 0$ $h_2(\vec{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$ $h_3(\vec{x}) = x_3 + x_6 - x_5 = 0$ $h_4(\vec{x}) = x_4 + x_7 - x_8 = 0$	$0 \leq x_1, x_2, x_6 \leq 300$ $0 \leq x_3, x_5, x_7 \leq 100$ $0 \leq x_4, x_8 \leq 200$ $0.01 \leq x_9 \leq 0.03$
G24	$f(\vec{x}) = -x_1 - x_2$	$g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$ $g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$	$0 \leq x_1 \leq 3$ $0 \leq x_2 \leq 4$

CEC 2010 Functions

Name	Function	Constraints	Bounds
C01	$f(x) = - \left \frac{\sum_{i=1}^D \cos^4(z_i) - 2 \prod_{i=1}^D \cos^2(z_i)}{\sqrt{\sum_{i=1}^D iz_i^2}} \right $ $z = x - 0$	$g_1(x) = 0.75 - \prod_{i=1}^D z_i \leq 0$ $g_2(x) = \sum_{i=1}^D z_i - 7.5D \leq 0$	$x \in [0, 10]^D$
C02	$f(x) = \max(z) \quad z = x - 0, y = z - 0.5$	$g_1(x) = 10 - \frac{1}{D} \sum_{i=1}^D [z_i^2 - 10 \cos(2\pi z_i) + 10] \leq 0$ $g_2(x) = \frac{1}{D} \sum_{i=1}^D [z_i^2 - 10 \cos(2\pi z_i) + 10] - 15 \leq 0$ $h_1(x) = \frac{1}{D} \sum_{i=1}^D [y_i^2 - 10 \cos(2\pi y_i) + 10] - 20 = 0$	$x \in [-5.12, 5.12]^D$
C03	$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) \quad z = x - 0$	$h_1(x) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2 = 0$	$x \in [-1000, 1000]^D$
C04	$f(x) = \max(z) \quad z = x - 0$	$h_1(x) = \frac{1}{D} \sum_{i=1}^D (z_i \cos(\sqrt{ z_i })) = 0$ $h_2(x) = \sum_{i=1}^{D/2-1} (z_i - z_{i+1})^2 = 0$ $h_3(x) = \sum_{i=D/2+1}^{D-1} (z_i^2 - z_{i+1})^2 = 0$ $h_4(x) = \sum_{i=1}^D z = 0$	$x \in [-50, 50]^D$
C05	$f(x) = \max(z) \quad z = x - 0$	$h_1(x) = \frac{1}{D} \sum_{i=1}^D (-z_i \sin(\sqrt{ z_i })) = 0$ $h_2(x) = \sum_{i=1}^D (-z_i \cos(0.5\sqrt{ z_i })) = 0$	$x \in [-600, 600]^D$

Name	Function	Constraints	Bounds
C06	$f(x) = \max(z)$ $z = x - o, y$ $= (x + 483.6106156535$ $- o)M - 483.6106156535$	$h_1(x) = \frac{1}{D} \sum_{i=1}^D (-y_i \sin(\sqrt{ y_i })) = 0$ $h_2(x) = \frac{1}{D} \sum_{i=1}^D (-y_i \cos(0.5\sqrt{ y_i })) = 0$	$x \in [-600, 600]^D$
C07	$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$ $z = x + 1 - o, y = x - o$	$g_1(x) = 0.5 - \exp\left(-0.1 \sqrt{\frac{1}{D} \sum_{i=1}^D y_i^2}\right) - 3 \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(0.1y)\right) + \exp(1) \leq 0$	$x \in [-140, 140]^D$
C08	$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$ $z = x + 1 - o, y = (x - o)M$	$g_1(x) = 0.5 - \exp\left(-0.1 \sqrt{\frac{1}{D} \sum_{i=1}^D y_i^2}\right) - 3 \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(0.1y)\right) + \exp(1) \leq 0$	$x \in [-140, 140]^D$
C09	$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$ $z = x + 1 - o, y = x - o$	$h_1(x) = \sum_{i=1}^D (y_i \sin(\sqrt{ y_i })) = 0$	$x \in [-500, 500]^D$
C10	$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$ $z = x + 1 - o, y = (x - o)M$	$h_1(x) = \sum_{i=1}^D (y_i \sin(\sqrt{ y_i })) = 0$	$x \in [-500, 500]^D$
C11	$f(x) = \frac{1}{D} \sum_{i=1}^D (-z_i \cos(2\sqrt{ z_i }))$ $z = (x - o)M, y = x + 1 - o$	$h_1(x) = \sum_{i=1}^{D-1} (100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2) = 0$	$x \in [-100, 100]^D$
C12	$f(x) = \sum_{i=1}^D (z_i \sin(\sqrt{ z_i }))$ $z = x - o$	$h_1(x) = \sum_{i=1}^{D-1} (z_i^2 - z_{i+1})^2 = 0$ $g_1(x) = \sum_{i=1}^D (z - 100 \cos(0.1z) + 10) \leq 0$	$x \in [-1000, 1000]^D$
C13	$f(x) = \frac{1}{D} \sum_{i=1}^D (-z_i \sin(\sqrt{ z_i }))$ $z = x - o$	$g_1(x) = -50 + \frac{1}{100D} \sum_{i=1}^D z_i^2 \leq 0$ $g_2(x) = \frac{50}{D} \sum_{i=1}^D \sin\left(\frac{1}{50} \pi z\right) \leq 0$ $g_3(x) = 75 - 50 \left(\sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 \right) \leq 0$	$x \in [-500, 500]^D$

Name	Function	Constraints	Bounds
C14	$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$ $z = x + 1 - o, y = x - o$	$g_1(x) = \sum_{i=1}^D (-y_i \cos(\sqrt{ y_i })) - D \leq 0$ $g_2(x) = \sum_{i=1}^D (y_i \cos(\sqrt{ y_i })) - D \leq 0$ $g_3(x) = \sum_{i=1}^D (y_i \sin(\sqrt{ y_i })) - 10D \leq 0$	$x \in [-1000, 1000]^D$
C15	$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$ $z = x + 1 - o, y = (x - o)M$	$g_1(x) = \sum_{i=1}^D (-y_i \cos(\sqrt{ y_i })) - D \leq 0$ $g_2(x) = \sum_{i=1}^D (y_i \cos(\sqrt{ y_i })) - D \leq 0$ $g_3(x) = \sum_{i=1}^D (y_i \sin(\sqrt{ y_i })) - 10D \leq 0$	$x \in [-1000, 1000]^D$
C16	$f(x) = \sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$ $z = x - o$	$g_1(x) = \sum_{i=1}^D [z_i^2 - 100 \cos(\pi z_i) + 10] \leq 0$ $g_2(x) = \prod_{i=1}^D z_i \leq 0$ $h_1(x) = \sum_{i=1}^D (z_i \sin(\sqrt{ z_i })) \leq 0$ $h_2(x) = \sum_{i=1}^D (-z_i \sin(\sqrt{ z_i })) \leq 0$	$x \in [-10, 10]^D$
C17	$f(x) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2 \quad z = x - o$	$g_1(x) = \prod_{i=1}^D z_i \leq 0 \quad g_2(x) = \sum_{i=1}^D z_i \leq 0$ $h_1(x) = \sum_{i=1}^D (z_i \sin(4\sqrt{ z_i })) \leq 0$	$x \in [-10, 10]^D$
C18	$f(x) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2 \quad z = x - o$	$g_1(x) = \frac{1}{D} \sum_{i=1}^D (-z_i \sin(\sqrt{ z_i })) \leq 0$ $h_1(x) = \frac{1}{D} \sum_{i=1}^D (z_i \sin(\sqrt{ z_i })) \leq 0$	$x \in [-50, 50]^D$