APRIMORAMENTO DA METAHEURÍSTICA *MULTIOBJECTIVE HARMONY SEARCH* PARA PROBLEMAS COM VARIÁVEIS CONTÍNUAS

Diego Geraldo
Instituto Tecnológico de Aeronáutica – ITA
Pça Mal. Eduardo Gomes, 50. 12228-900, São José dos Campos, SP
diegog@ita.br

Mônica Maria de Marchi
Instituto de Estudos Avançados – IEAV
Rodovia dos Tamoios, km 5,5. 12228-001, São José dos Campos, SP
monica@ieav.cta.br

RESUMO

Este artigo tem por objetivo propor melhorias relacionadas aos principais operadores de busca e de diversidade do algoritmo *Multiobjective Harmony Search* (MOHS) e testar a efetividade dessas mudanças de forma a tornar o algoritmo mais adequado às características do ambiente multiobjetivo e, em particular, aos problemas de variáveis contínuas. A validade das sugestões foi verificada por meio da aplicação da versão aprimorada do algoritmo ao conjunto de funções-teste ZDT, e o seu desempenho foi mensurado com base nas métricas *Generational Distance* e *Spread*. Os resultados foram comparados aos obtidos pelo algoritmo *Nondominated Sorting Genetic Algorithm* II (NSGA-II). A análise estatística dos resultados permite afirmar que o desempenho do algoritmo aprimorado, denominado SA_MOHS, foi superior ao do NSGA-II.

PALAVRAS CHAVE. Otimização, Multiobjetivo, Busca Harmônica.

Computação Evolutiva, Metaheurísticas

ABSTRACT

This paper proposes some improvements in the Multiobjective Harmony Search Algorithm (MOHS) in relation to the search operator and the diversity operator. The purpose of this is to have an algorithm more suitable to the multiobjective characteristics and, in particular, to problems with continuous variables. Our approach was tested on ZDT function tests and the performance was measured based on metrics Generational Distance and Spread. The results were compared to those obtained by Nondominated Sorting Genetic Algorithm II (NSGA-II). The statistical analysis results suggests that the improved algorithm performance, called SA_MOHS, was superior to that of NSGA-II.

KEYWORDS. Optimization, Multiobjective, Harmony Search, Metaheuristics.

Evolutionary Computation, Metaheuristics

1. Introdução

Muitos problemas reais de engenharia são considerados, por natureza, multiobjetivo. Nos casos em que dois ou mais objetivos conflitantes passam a ser analisados simultaneamente, a melhora em um dos critérios resultará na piora de pelo menos um dos outros objetivos. Diferentemente dos problemas de otimização mono-objetivo, para os quais, em geral, busca-se encontrar apenas uma solução ótima, no campo da otimização multiobjetivo, o que se busca é encontrar um conjunto de soluções ótimas, que representem as diversas situações de compromisso à disposição de um decisor.

Ao longo dos anos, a comunidade de Pesquisa Operacional produziu muitas técnicas de programação matemática para lidar com os problemas multiobjetivo (Coello Coello, 2001). No entanto, as técnicas de programação matemática apresentam algumas limitações quando empregadas na abordagem de problemas com múltiplos critérios, podendo ser ineficazes nas situações em que a Fronteira de Pareto seja côncava ou descontínua, além de, na maioria das vezes, gerar apenas uma solução a cada execução do código (Chen e Lu, 2008).

Algoritmos evolutivos, por outro lado, são considerados mais adequados à resolução de problemas multiobjetivo porquanto possuem mecanismos capazes de lidar com uma população de soluções simultaneamente, o que lhes permite encontrar várias soluções em uma única execução do algoritmo (Deb, 2001).

Neste trabalho aplica-se o algoritmo *Multiobjective Harmony Search* (MOHS), baseado na metaheurística *Harmony Search* (HS) e que faz uso de mecanismos de preservação de diversidade e elitismo, bem como de dominância de Pareto. Ricart *et al*. (2011) testaram este algoritmo em um conjunto de funções-teste bastante comum na literatura, denominado ZDT, de seus autores Zitzler, Deb Thiele (2000), e os resultados foram comparados aos obtidos pelo algoritmo *Nondominated Sorting Genetic Algorithm* II (NSGA-II) (Deb *et al.*, 2002). Segundo estes autores, os resultados obtidos permitiram afirmar que o MOHS mostrou-se competitivo frente ao NSGA-II.

Entretanto, os principais operadores de busca do MOHS foram mantidos idênticos aos utilizados pela versão mono-objetivo do algoritmo, sendo passíveis, portanto, de aprimoramentos que os tornem mais adequados às características do ambiente multiobjetivo destinado à resolução de problemas que tenham variáveis contínuas na formulação.

Nesse sentido, o presente trabalho tem por objetivo: propor melhorias relacionadas aos principais operadores de busca e de diversidade do MOHS, bem como testar a efetividade dessas mudanças.

O restante do artigo está organizado da seguinte forma: a Seção 2 oferece uma breve introdução à otimização multiobjetivo; na Seção 3, a metaheurística HS e sua versão multiobjetivo são apresentadas; na Seção 4, as sugestões de aprimoramento ao MOHS e a argumentação teórica são discutidas, a Seção 5 apresenta os resultados dos testes e a análise estatística dos mesmos; a Seção 6, por fim, contempla as conclusões do trabalho.

2. Otimização Multiobjetivo

Formalmente, podemos definir um problema de otimização multiobjetivo como sendo:

$$\begin{aligned} &\textbf{Minimizar/Maximizar:} && y = F(x) = [f_1(x), f_2(x), \ldots, f_k(x)] \\ &\textbf{Em que :} && y = [y_1, y_2, \ldots, y_k] \in Y \subseteq R^k \\ & && x = [x_1, x_2, \ldots, x_n] \in X \subseteq R^n \\ & && x_i^{(L)} \leq x_i \leq x_i^{(U)} \end{aligned} \qquad (1)$$

Em (1), x é um vetor *n*-dimensional de variáveis de decisão sujeitas a restrições laterais inferior ($x_i^{(L)}$) e superior ($x_i^{(U)}$), y é um vetor *k*-dimensional de objetivos, $X \subseteq R^n$ denota o espaço de variáveis do problema e $Y \subseteq R^k$ representa o espaço de objetivos do problema. O restante desse artigo assume a minimização de *k* funções-objetivo.

Em muitos problemas reais de otimização, os objetivos sendo considerados podem ser conflitantes entre si. Dessa maneira, otimizar uma solução $x \in X \subseteq R^n$ com relação a um dos objetivos apenas, pode resultar, na maioria das vezes, em uma solução inaceitável em relação aos demais objetivos. Assim, a obtenção de uma única solução que otimize simultaneamente cada uma das funções é praticamente impossível (Konak, Coit and Smith, 2006).

Uma resposta razoável para um problema multiobjetivo é obter um conjunto de soluções de compromisso, de maneira que cada uma delas atenda aos objetivos em níveis aceitáveis (Zitzler, Laumanns and Bleuler, 2004). A existência de mais de uma solução ótima em problemas multiobjetivo leva à necessidade de uma definição diferente de otimalidade. Coello Coello (2001) afirma que a definição mais comumente utilizada nesse sentido é a de 'Pareto-ótimo'.

Segundo o conceito de dominância de Pareto, um vetor de objetivos y^1 domina outro vetor de objetivos y^2 ($y^1 \prec y^2$) se nenhum dos componentes de y^1 for maior do que seu correspondente em y^2, e se pelo menos um dos elementos de y^1 for menor que o correspondente em y^2. Analogamente, podemos dizer que uma solução x^1 é melhor que outra solução x^2, i.e., x^1 domina x^2 ($x^1 \prec x^2$), se $f(x^1)$ dominar $f(x^2)$ (Zitzler, Laumanns and Bleuler, 2004).

O conjunto de soluções ótimas no espaço de variáveis é geralmente denominado Conjunto de Pareto, e a correspondente imagem no espaço objetivo, Fronteira de Pareto.

O principal objetivo dos algoritmos de otimização multiobjetivo é identificar as soluções pertencentes (ou próximas) ao Conjunto de Pareto. No entanto, para muitos problemas, o número de soluções Pareto ótimas pode ser infinito, o que torna inviável, ou mesmo impossível, a consecução de tal tarefa. Portanto, partindo de uma abordagem mais prática, podemos assumir que os algoritmos multiobjetivo buscam identificar um conjunto finito de soluções que representem o Conjunto de Pareto tão bem quanto possível (Konak, Coit and Smith, 2006).

Segundo Ziztler *et al*. (2000), uma abordagem por otimização multiobjetivo deve visar à obtenção de três objetivos conflitantes, descritos a seguir:

- a Fronteira de Pareto aproximada obtida pelo otimizador deve estar o mais próximo possível Fronteira de Pareto real;

- as soluções obtidas devem estar uniformemente distribuídas ao longo de toda a Fronteira de Pareto, a fim de permitir ao decisor uma visão completa e abrangente das melhores relações de compromisso entre os diversos objetivos;

- a fronteira obtida pelo otimizador deve capturar toda a gama da Fronteira de Pareto real. Isso requer exploração dos extremos do espaço de objetivos.

O primeiro objetivo pode ser alcançado através da intensificação da busca em uma região específica da fronteira, porém, o segundo objetivo requer que os esforços estejam igualmente distribuídos ao longo da mesma, enquanto que o terceiro objetivo visa estender a fronteira, explorando soluções em seus extremos.

3. A metaheurística *Harmony Search*

A metaheurística denominada *Harmony Search* (Geem, Kim e Loganathan, 2001) é um mecanismo de busca global inspirada no processo de improvisação musical. A HS foi proposta inicialmente como uma metaheurística para otimização de problemas mono-objetivo e seus mecanismos podem ser mais bem compreendidos se comparados ao processo de improvisação realizado por um músico.

Quando um músico está improvisando, ele escolhe basicamente dentre três opções: (1) executar qualquer trecho de uma música conhecida exatamente como está em sua memória; (2) executar qualquer trecho de uma música conhecida realizando pequenos ajustes ou adaptações; ou (3) compor novos tons aleatoriamente (Yang, 2008). Ao realizar uma sequência dessas escolhas, o músico busca estabelecer um estado perfeito de harmonia, segundo os padrões da estética musical.

Geem, Kim e Loganathan (2001) formalizaram essas três regras em um algoritmo de otimização, em que o valor de uma variável de decisão pode ser atribuído de três maneiras

distintas: (1) assumindo qualquer um dos valores já existentes em memória; (2) assumindo um valor ligeiramente diferente de qualquer outro existente em memória; ou (3) assumindo um valor aleatório dentre todo o intervalo permitido. Essas três opções foram operacionalizadas no algoritmo por meio dos seguintes parâmetros livres: *harmony memory size* (HMS), *harmony memory accepting* (HMA), *pitch-adjusting rate* (PAR) e *pitch range variability* (PRV).

O parâmetro HMS representa o número de soluções presentes na memória, similarmente ao número de indivíduos na população de um Algoritmo Genético.

O parâmetro HMA, por sua vez, pode ser visto como a probabilidade de atribuir a uma variável, um valor dentre os já existentes em memória. O correto ajuste deste parâmetro garante que as boas soluções sejam consideradas na composição de novas soluções. Tipicamente, HMA $\in$ [0.7, 0.95] (Yang, 2008).

O mecanismo de *pitch adjusting* pode ser equiparado ao operador de mutação de um Algoritmo Genético. No entanto, diferentemente deste último, o primeiro visa explorar o espaço de busca localmente, por meio da adição ou subtração de pequenas perturbações às variáveis. O parâmetro PAR controla a frequência com que essas perturbações são aplicadas, e o PRV, por sua vez, a magnitude das mesmas. O PAR é, normalmente, ajustado com valores pertencentes ao intervalo [0.1, 0.5], e o PRV é limitado a valores entre 1% e 10% de todo o intervalo permitido para as variáveis (Yang, 2008).

O mecanismo responsável por gerar novos valores aleatórios para as variáveis tem por função prover a diversificação das soluções e melhorar a exploração do espaço de busca.

Os três principais componentes da metaheurística *Harmony Search*, quais sejam, o uso da memória, perturbações locais, e aleatoriedade, podem ser identificadas no pseudocódigo da Figura 1, em que *rand* denota um valor amostrado de uma distribuição de probabilidades Uniforme no intervalo especificado.

Inputs: *f*(x), HMS, HMA, PAR, PRV.
Output: x^{melhor} armazenado na HM.
Inicialize aleatoriamente HM com HMS soluções
while (critério de parada não satisfeito) **do**
  for (cada solução *x*) **do**
    for (cada variável x_i) **do**
      if (*rand* [0, 1] < HMA) **then**
        $x_i' = x_i^j$, em que j = int(*rand* [1, HMS])
        if (*rand* [0,1] < PAR) **then**
          $x_i' = x_i' + rand\ [-1, 1] * PRV$
        end if
      else
        x_i' = valor aleatório
      end if
    end for
  end for
  if (*x'* melhor do que x^{pior} na HM) **then**
    Substitua x^{pior} por *x'*
  end if
end while

Figura 1. Pseudocódigo da metaheurística *Harmony Search*. Adaptado de Ricart *et al.* (2011).

No pseudocódigo da Figura 1, observamos que a probabilidade de atribuir um valor aleatório a uma variável é de (1 - HMA). A probabilidade de um valor já existente na memória ser atribuído à variável é de HMA*(1 - PAR), e a probabilidade desse valor ser perturbado localmente é de (HMA*PAR).

3.1 Versões multiobjetivo da metaheurística *Harmony Search*

Recentemente, duas versões multiobjetivo da metaheurística HS foram propostas (Ricart *et al*., 2011). No artigo em questão, os autores argumentam que apesar de existir na literatura algumas aplicações da HS em problemas de otimização multiobjetivo, na maioria delas, o conceito de otimalidade de Pareto não foi explicitamente utilizado. Ainda segundo os autores, em algumas referências, o conceito Pareto-ótimo chegou a ser aplicado, porém, sem qualquer detalhamento a respeito das modificações realizadas no algoritmo original a fim de se adaptar às peculiaridades da otimização multiobjetivo. O leitor interessado em uma breve revisão a respeito de aplicações da HS em aplicações multiobjetivo deve se referir à Ricart *et al*. (2011).

Em ambas as propostas, denominadas MOHS1 e MOHS2, a HM é utilizada como um repositório das soluções que representam as melhores relações de compromisso encontradas pelo algoritmo em um dado instante da busca, não sendo utilizados, portanto, "arquivos externos" nas implementações. Dessa forma, as soluções não-dominadas presentes na HM representam uma aproximação do Conjunto de Pareto do problema abordado. Além disso, os autores utilizaram o procedimento proposto por Fonseca e Flemming (1993) para mensurar a qualidade das soluções e tornar possível o ranqueamento das mesmas.

A primeira proposta, em particular, foi concebida de modo a não requerer mudanças significativas no comportamento da *Harmony Search* original (Ricart *et al*., 2011). Nesta implementação, uma nova solução é gerada a cada iteração e a mesma é aceita na HM se seu *rank* resultante for melhor que o *rank* da pior solução presente na memória. Nenhum mecanismo de preservação de diversidade é explicitamente empregado nessa versão.

A segunda proposta, ao contrário da primeira, prevê modificações no *modus operandi* da metaheurística original. A cada iteração, uma nova memória HM2, de tamanho idêntico à memória original, HM1, é gerada com base nas soluções contidas nesta última. Da união de ambas as memórias (HM1 $\cup$ HM2), apenas metade das soluções são admitidas como componentes da memória da próxima geração. No MOHS2, o mecanismo de seleção das soluções compreende a atribuição de *fitness* por *ranking* de Fonseca-Flemming, e um método de preservação de diversidade, denominado *Truncate Procedure*, similar ao empregado no algoritmo SPEA2 (Zitzler, Laumanns e Thiele, 2001). O pseudocódigo do MOHS2 é exibido na Figura 2.

Inputs: F(x), HMS, HMA, PAR, PRV.
Output: Conjunto Pareto-ótimo armazenado em HM1.
Inicialize aleatoriamente HM1 com HMS soluções
while (critério de parada não satisfeito) **do**
    Inicialize HM2 vazia, de tamanho idêntico a HM1
    while (HM2 não completamente preenchida) **do**
        gere uma nova solução S com base em HM1
        armazene S em HM2
    end while
    Defina HM_u = HM1 $\cup$ HM2
    Ranking das soluções de HM_u com base no método de Fonseca e Fleming
    Defina HM1 vazia
    r = 1
    F = soluções em HM_u com rank = r
    while (HM1 não completamente preenchida) **do**
        if (houver espaço em HM1 para acomodar F) **then**
            mova as soluções de F para HM1
            r = r + 1
        else
            Truncate Procedure em F
        end if
    end while
end while

Figura 2. Pseudocódigo do algoritmo *Multiobjective Harmony Search* 2 (MOHS2). Adaptado de Ricart *et al.* (2011).

Segundo Deb (2001), um algoritmo de otimização multiobjetivo será tão melhor quanto maior forem suas capacidades de (i) convergir para aquilo que se espera ser a Fronteira de Pareto real, e (ii) manter um espalhamento adequado das soluções ao longo da mesma. Assim, nossas propostas de aprimoramento centralizam-se apenas na segunda versão, MOHS2, porquanto tal implementação prevê em sua estrutura, mecanismos capazes de atender a esses dois requisitos simultaneamente. Por simplificação, passaremos a nos referir ao MOHS2 apenas pelo termo abreviado MOHS, e à versão aprimorada, denominaremos SA_MOHS (Sugestões de Aprimoramento para o MOHS).

4. Sugestões de aprimoramento para o MOHS (SA_MOHS)

Nossas sugestões de aprimoramento compreendem, basicamente, (i) a substituição do mecanismo de aleatoriedade do algoritmo original, por outro menos disruptivo, e (ii) a substituição do mecanismo de perturbação local das variáveis por outro mais sofisticado. Adicionalmente, substituímos o mecanismo de atribuição de *fitness* às soluções por outro que requer, *a priori*, esforço computacional similar ou menor ao proposto na versão original. Apresentamos, a seguir, as justificativas e os detalhes para cada uma das proposições.

4.1 Aprimoramento relacionado ao operador de aleatoriedade

Na versão mono-objetivo do HS, novas soluções são inseridas uma a uma na memória, e uma nova solução somente é aceita como parte da memória se for melhor do que a pior já existente. Este mecanismo de preservação de elitismo atribui uma característica de pressão de seleção inerentemente alta ao algoritmo, o que leva à necessidade de existir outro mecanismo, igualmente agressivo, que seja capaz de manter a diversidade entre as soluções, evitando assim, a convergência prematura (Goldberg e Deb, 1991 *apud* Deb, 2001). O operador de diversidade do HS, definido por meio do parâmetro HMA, atua exatamente nesse sentido.

Entretanto, argumentamos que, no contexto da otimização multiobjetivo, esse mecanismo de aleatoriedade pode ser excessivamente destrutivo, chegando mesmo a impedir por completo a convergência do algoritmo. Isso se deve, principalmente, ao fato do atual operador de diversidade não levar em consideração nenhuma característica das soluções já existentes na memória ao gerar novos valores para as variáveis.

Deb (2001) estabelece uma importante analogia entre otimização mono-objetivo de funções multimodais e otimização multiobjetivo, ambas utilizando Algoritmos Genéticos, onde o que se deseja, é a obtenção de diversos ótimos simultaneamente. Para alcançar esse objetivo, é necessário que as boas soluções, aquelas que representem os possíveis ótimos do espaço de busca, sejam mantidas na população durante muitas gerações. Nesse sentido, operadores de diversidade que não possuam características autoadaptativas, podem acabar atuando mais como "destrutores" de boas soluções, do que como construtores delas (Deb, 2001).

Com essa discussão, queremos dizer que o desempenho do algoritmo original pode se tornar extremamente sensível ao correto ajuste do parâmetro HMA. Como as probabilidades de 'considerar um valor já existente' e 'sortear um valor aleatório' são complementares, se definirmos, por exemplo, o valor de HMA em 0,7 (70%), espera-se que cerca de 30% das variáveis de uma determinada solução tenham seus valores modificados por um operador de diversidade com características destrutivas, sob o ponto de vista da otimização multiobjetivo.

Nossa proposta de aprimoramento compreende a substituição do atual operador de diversidade pelo *crossover* BLX-α, no qual uma nova solução é gerada a partir de duas outras soluções já existentes (Goldberg, 1991 *apud* Deb, 2001), segundo a equação:

$$x_i' = x_i^1 + \beta(x_i^2 - x_i^1) \qquad (2)$$

Em (2), β é um número aleatório proveniente de uma distribuição uniforme no intervalo [-α, 1+α]. Quando α = 0, o operador gera uma nova solução aleatória compreendida no intervalo $[x_i^1, x_i^2]$, e quando α > 0, esse intervalo é estendido para além dos limites definidos pelas posições das variáveis originais. Neste trabalho, α foi implementado com valor 0,25.

A justificativa para o uso do BLX-α é bastante simples. No *crossover* BLX, a localização da nova solução gerada (filho) depende da diferença de localização das soluções originais (pais). Assim, se a diferença entre os pais for pequena, a diferença entre o filho e os pais também será pequena, sendo o oposto, também verdadeiro. Esta importante propriedade atribui ao operador a desejável característica de exploração adaptativa do espaço de busca.

Acreditamos que a substituição do operador de diversidade proposto originalmente pelo BLX torne o algoritmo menos suscetível e, portanto, mais robusto, ao ajuste do parâmetro HMA.

4.2 Aprimoramento relacionado ao operador de perturbação local das variáveis

O operador de perturbação local originalmente proposto no MOHS atribui a uma variável um novo valor proveniente de sua própria vizinhança, escolhido com distribuição de probabilidade Uniforme. Para tanto, é necessário que se definam os limites máximos dessa vizinhança.

Nossa proposta de melhoria envolve a substituição desse operador, baseado em distribuição de probabilidade Uniforme, por outro mais sofisticado, conhecido na literatura por operador de mutação polinomial (Deb e Goyal, 1996), no qual a distribuição de probabilidade é definida por uma função polinomial, que tem o valor atual da variável como média, e variância sendo diretamente controlada por um parâmetro η. Segundo Deb (2001), um valor η produz uma perturbação de ordem $O(1/\eta)$ em uma variável de decisão normalizada.

A formulação da mutação polinomial não será aqui discutida e o leitor interessado deve se referir à Deb e Goyal (1996) para maiores detalhes.

A fim de tornar mais clara a diferença entre os operadores citados, realizamos a perturbação em uma variável $x = 5$, com limites laterais definidos pelo intervalo [0,10], cem mil vezes (10^5) com cada operador, e em seguida, construímos os histogramas dos resultados, que representam a distribuição de probabilidade aproximada dos valores atribuídos à variável *x* após a perturbação da mesma. Utilizamos os valores de η = 10 para o operador de mutação polinomial, e 10% do intervalo definido pelos limites laterais de *x* como limite máximo de perturbação para o operador de mutação uniforme. Os gráficos são apresentados na Figura 3.

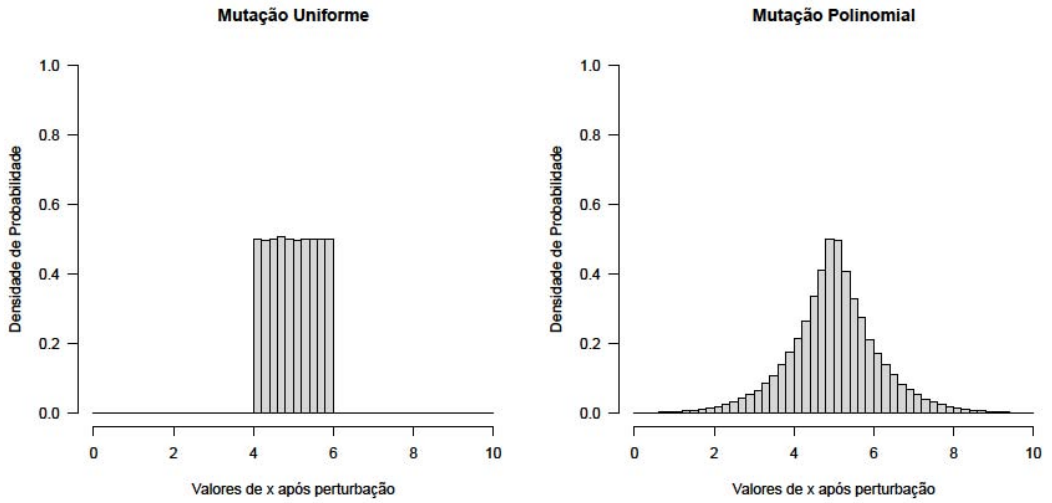

Figura 3. Distribuição de probabilidade empírica dos valores atribuídos à variável *x* após a perturbação, obtidas a partir dos operadores de mutação uniforme e polinomial.

A análise dos gráficos da Figura 3 mostra que as perturbações geradas pelo operador de mutação polinomial resultam em uma distribuição de probabilidade caracterizada por caudas e por um pico nas proximidades do atual valor da variável, o que pode ser vantajoso para o algoritmo em dois aspectos: (i) intensificação da busca local nas vizinhanças imediatas da variável, (ii) menor susceptibilidade aos atrativos locais, uma vez que perturbações de maior magnitude têm probabilidade não desprezível de ocorrerem.

Além disso, o operador polinomial possui mecanismos em sua formulação que lidam com variáveis sujeitas a restrições laterais, ao passo que, ao se utilizar o operador uniforme, soluções inviáveis poderão ser geradas (Deb, 2001). No artigo original do MOHS os autores não tornaram explícito o método utilizado para lidar com as soluções inviáveis geradas a partir das perturbações.

Neste trabalho, o operador de mutação polinomial foi implementado tal qual proposto em Deb e Goyal (1996), e o valor de $\eta = 30$ foi utilizado em todos os experimentos.

4.3 Discussões a respeito da atribuição de *fitness* às soluções

Uma das principais dificuldades relacionadas ao uso de algoritmos evolutivos na otimização de problemas multiobjetivo diz respeito à atribuição de um único valor de *fitness* a uma determinada solução, em presença dos diversos valores das funções-objetivo resultantes da mesma.

Diferentes abordagens baseadas no conceito de dominância de Pareto foram propostas como alternativa em face dessa dificuldade. Nessas abordagens, a população de soluções é ranqueada de acordo com o conceito de dominância e o *fitness* de cada solução passa a ser dado por seu *rank*, em substituição às abordagens clássicas que se valem de métodos agregadores dos valores das funções-objetivo propriamente ditos.

Duas das principais técnicas de ranqueamento propostas na literatura foram a de Goldberg (1989) e a de Fonseca e Fleming (1993). Considerando essas duas abordagens, nenhuma parece, a *priori*, ser superior à outra. Entretanto, há reportes na literatura que destacam o fato do método de Fonseca e Fleming proporcionar maior número de *ranks*, possuindo, portanto, maior capacidade discricionária do que a proposta de Goldberg (assumindo uma população de tamanho fixo) (Coello Coello, Lamont e Van Veldhuizen, 2007). Além disso, o método de Fonseca e Fleming penaliza as soluções dominadas por seções densamente povoadas da Fronteira de Pareto no espaço de objetivos (Konak, Coit e Smith, 2006).

Deb (2001) argumenta que no método de Fonseca e Fleming, nem todas as soluções pertencentes a uma mesma fronteira (exceto a primeira) possuem valor de *fitness* comum, o que pode introduzir um viés indesejável em direção a algumas soluções do espaço de busca. O autor alega ainda que o algoritmo *Multiobjetive Genetic Algorithm* (MOGA), que faz uso desse método de ranqueamento, em particular, pode ser sensível ao formato da Fronteira de Pareto e à densidade de soluções no espaço de busca.

Com relação à ordem de complexidade computacional dos métodos citados, ambos requerem $O(\mathrm{kN}^2)$ comparações, onde k representa o número de objetivos e N, o número de indivíduos na população (Deb, 2001).

A atribuição de *fitness* às soluções baseados nos métodos de ranqueamento estão relacionados diretamente à capacidade de convergência dos algoritmos. A fim de alcançar o segundo objetivo da otimização multiobjetivo (diversidade de soluções ao longo da Fronteira de Pareto), alguns mecanismos de preservação de diversidade foram propostos na literatura, com o intuito de dar ênfase às soluções localizadas em regiões menos "povoadas" do espaço de busca.

Dois dentre os principais mecanismos propostos na literatura para esse fim são o *Crowding Distance Assignment* (Deb *et al.*, 2002), e o *Truncate Procedure* (Zitzler, Laumanns e Thiele, 2001). No primeiro deles, uma estimativa da densidade de soluções ao redor de uma solução x, em particular, é dada pelo perímetro do cuboide formado usando-se os vizinhos mais próximos como vértices. No segundo método citado, um procedimento iterativo exclui da população, a cada ciclo, o indivíduo que possuir a menor distância em relação às outras soluções, até que a população atinja o tamanho desejado.

Ambos os métodos apresentam a vantagem de não necessitarem de parâmetros a ajustar. Com relação à ordem de complexidade computacional dos algoritmos, o *Crowding Distance Assignment* requer $O(k\mathrm{N}\log\mathrm{N})$ comparações (Deb *et al.*, 2002), enquanto o *Truncate Procedure* necessita, em média, $O(\mathrm{N}^2\log\mathrm{N})$ comparações, podendo chegar a $O(\mathrm{N}^3)$ na pior das situações (Zitzler, Laumanns e Thiele, 2001).

Em função da menor ordem de complexidade computacional escolhemos o *Crowding Distance Assignment* como mecanismo de manutenção de diversidade em nossa implementação, não significando, no entanto, que tal escolha constitua necessariamente uma sugestão de aprimoramento. Além disso, como não encontramos na literatura uma definição clara a respeito da superioridade de um método de ranqueamento sobre outro, optamos por substituir a técnica de Fonseca e Fleming utilizada na proposta original, pela de Goldberg, uma vez que a métrica de *crowding distance* é utilizada como critério de desempate nos casos em que duas soluções sendo comparadas segundo o critério de não-dominância pertencem a uma mesma fronteira.

5. Experimentos e Resultados

A fim de testar a validade das sugestões de aprimoramento e permitir a comparação entre algoritmos, elegemos cinco das seis funções-teste conhecidas como ZDT (Zitzler, Deb e Thiele, 2000). As funções ZDT1, ZDT2, ZDT3, ZDT4 e ZDT6 contemplam características como convexidade, não convexidade, descontinuidade, multimodalidade e não uniformidade (Deb, 2001). Os problemas assumem a minimização irrestrita de duas funções-objetivo e variáveis de decisão contínuas. O problema ZDT5 contempla variáveis discretas com codificação binária e, portanto, não será objeto de estudo deste trabalho.

Em Ricart *et al.* (2011), o MOHS foi executado 10 vezes durante 10 segundos, para cada uma das funções ZDT, e os resultados foram comparados aos obtidos pelo algoritmo NSGA-II (Deb *et al.*, 2002). Em nosso trabalho, mantivemos a comparação de resultados da versão aprimorada com o NSGA-II, porém, utilizamos um critério de parada que independente de características de *hardware* e de linguagem de programação, mais comumente utilizado na literatura, limitado a 25.000 avaliações das funções-objetivo (Deb *et al.*, 2002; Zitzler, Deb e Thiele, 2000; Chen e Lu, 2008).

Os algoritmos foram comparados com base em duas métricas de desempenho conhecidas na literatura: *Generational Distance* (GD) e *Spread* (SPD), tal qual descritas em Deb (2001). O computo das métricas foi realizado por meio do pacote *mco* (Trautmann, Steuer e Mersmann, 2010) do *software* R (R Development Core Team, 2011).

A primeira métrica citada oferece uma estimativa da capacidade de convergência do algoritmo, medindo o quanto as soluções obtidas pelo algoritmo se aproximam da real Fronteira de Pareto. Portanto, quanto menor o valor dessa métrica, melhor o desempenho do algoritmo. A segunda métrica, por sua vez, estima a uniformidade de espaçamento entre as soluções obtidas, bem como a porção da Fronteira Pareto-ótima capturada pelo algoritmo. Da mesma maneira, valores menores para essa métrica sugerem melhor desempenho.

O SA_MOHS foi implementado em linguagem R e, por simplicidade, utilizamos o mesmo ajuste de parâmetros utilizado no MOHS em seu artigo original, ou seja, HMS = 100; HMA = 0,95 e PAR = 0,1. Reservamos a tarefa de ajuste de parâmetros do algoritmo para trabalhos futuros. O NSGA-II foi implementado por meio da função *'nsga2'* do pacote *'mco'* do *software* R. Os parâmetros do NSGA-II foram ajustados tal qual sugeridos por Deb *et al.* (2002), ou seja, tamanho da população *pop* = 100, probabilidade de *crossover* p_c = 0,9, probabilidade de mutação $p_m = 1/n$ (onde n é o número de variáveis do problema), índice de distribuição do *crossover* SBX $\eta_c = 20$, e índice de distribuição da mutação polinomial $\eta_m = 20$.

Cada algoritmo foi executado 30 vezes para cada um dos problemas com, no máximo, 25.000 avaliações das funções. Os resultados provenientes das simulações são apresentados nas Tabelas 1 (GD) e 2 (SPD). Essas tabelas mostram os valores médios e os desvios-padrão obtidos por cada algoritmo em cada função-teste. A Figura 4 fornece a representação gráfica dos resultados, onde os valores médios obtidos pelos algoritmos em cada função-teste e em cada métrica estão representados pelas barras, e os intervalos 2σ ($\mu + \sigma$, $\mu - \sigma$, com σ = desvio padrão) estão representados pelas barras de erro.

Tabela 1 – Médias (μ) e desvios-padrão (σ) da métrica *Generational Distance*.

Algoritmo		ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
SA_MOHS	μ	0.000100	0.000102	0.000077	0.000115	0.000118
	σ	0.000006	0.000007	0.000005	0.000030	0.000008
NSGA-II	μ	0.000202	0.000188	0.000111	0.000411	0.000864
	σ	0.000024	0.000029	0.000012	0.000159	0.000116

Tabela 2 – Médias (μ) e desvios-padrão (σ) da métrica *Spread*.

Algoritmo		ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
SA_MOHS	μ	0.000067	0.000010	0.000244	0.004711	0.001512
	σ	0.000148	0.000027	0.000222	0.003689	0.000308
NSGA II	μ	0.001740	0.002283	0.001796	0.010052	0.019973
	σ	0.000857	0.000417	0.000738	0.004550	0.003312

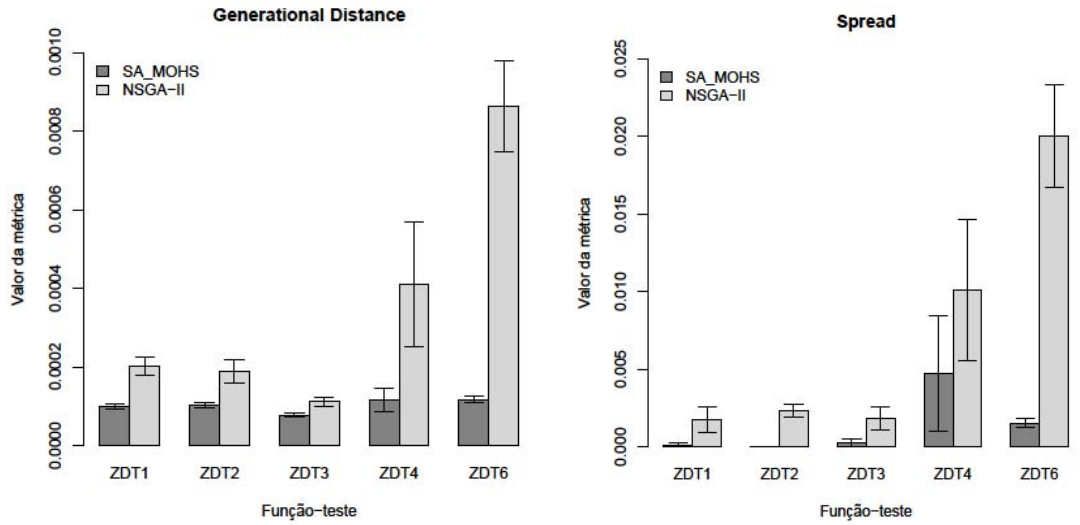

Figura 4. Médias (μ) e intervalos 2σ obtidos pelos algoritmos em cada métrica e em cada função-teste.

Os dados das Tabelas 1 e 2, bem como os gráficos da Figura 4, oferecem evidencias de que a performance do SA_MOHS nos testes realizados foi superior ao do NSGA-II.

No entanto, como ambos os algoritmos são regidos por mecanismos probabilísticos e como o número de amostras é relativamente pequeno, realizamos análises estatísticas mais criteriosas, utilizando o teste de hipóteses de Wilcoxon-Mann-Whitney para amostras independentes (Wonnacott e Wonnacott, 1990). Optamos por um teste da estatística não paramétrica uma vez que a análise exploratória dos dados, realizada por meio da construção de *boxplots*, evidenciou a existência de assimetrias e não normalidade nas distribuições de algumas amostras, bem como a existência de *outliers*. Os *boxplots* foram omitidos por restrições de espaço.

A hipótese nula assumida nos testes é de que diferença entre as medianas das duas populações é zero (“as performances são iguais”), e a hipótese alternativa é de que a diferença entre elas é maior que zero (“o desempenho do SA_MOHS é superior ao do NSGA-II”). As Tabelas 3 (GD) e 4 (SPD) exibem os resultados dos testes estatísticos.

Tabela 3 – Resultados dos testes de Wilcoxon-Mann-Whitney para a métrica *Generational Distance* (SA_MOHS x NSGA-II).

Função-Teste	Estatística W	Dif. em localização	Valor P	Lim Inf IC (95%)
ZDT1	900	0.000100	2.20E-16	0.000095
ZDT2	900	0.000087	2.20E-16	0.000074
ZDT3	900	0.000035	2.20E-16	0.000030
ZDT4	897	0.000264	2.20E-16	0.000212
ZDT6	900	0.000738	2.20E-16	0.000700

Tabela 4 – Resultados dos testes de Wilcoxon-Mann-Whitney para a métrica *Spread* (SA_MOHS x NSGA-II).

Função-Teste	Estatística W	Dif. em localização	Valor P	Lim Inf IC (95%)
ZDT1	900	0.001466	2.20E-16	0.001228
ZDT2	900	0.002222	2.20E-16	0.002159
ZDT3	898	0.001481	2.20E-16	0.001297
ZDT4	736	0.005321	5.10E-06	0.003476
ZDT6	900	0.018379	2.20E-16	0.017064

A análise exploratória dos dados amostrados e os testes estatísticos realizados indicam que as hipóteses de igualdade de desempenho entre SA_MOHS e NSGA-II devem ser rejeitadas em favor da hipótese alternativa, ou seja, o desempenho do SA_MOHS foi superior ao do NSGA-II, considerando o conjunto de problemas e as parametrizações utilizadas nos testes.

Das comparações dos resultados entre as duas versões do MOHS e o NSGA-II apresentados em Ricart *et al.* (2011), os autores concluiram que nenhum dos algoritmos (MOHS1, MOHS2 e NSGA-II) superou os outros para cada problema e em todas as métricas, mas os algoritmos MOHS mostraram-se competitivos quando comparados a uma alternativa "estado da arte" o NSGA-II.

Nossa versão aprimorada, i.e., SA_MOHS, não somente se mostrou competitivo em relação ao NSGA-II, como também obteve resultados melhores, estatisticamente comprovados, em todos os problemas e métricas avaliados. Ressaltamos também que nossos experimentos foram conduzidos com base em um número típico de avaliações das funções-objetivo (25.000), que independe de características de *hardware* e linguagem utilizada, e que tornam os resultados mais facilmente reproduzíveis.

6. Conclusões

Neste artigo, apresentamos sugestões de melhorias no algoritmo MOHS, que se resumiram basicamente em: (i) substituir o operador de diversidade com características disruptivas pelo *crossover* BLX-α, que possui mecanismos autoadaptativos; e (ii) substituir o operador de busca (perturbação) local, baseado em distribuição de probabilidade uniforme, por outro mais eficiente, baseado em distribuição de probabilidade polinomial. Nossas propostas visavam à melhoria da robustez e da eficiência do algoritmo, em particular, quando aplicado a problemas com variáveis contínuas.

A fim de validar a efetividade das alterações propostas, aplicamos o SA_MOHS ao conjunto de funções-teste ZDT e usamos duas métricas, *Generational Distance* e *Spread*, para mensurar o desempenho do algoritmo. Comparamos os resultados do SA_MOHS aos obtidos pelo já bem estabelecido algoritmo NSGA-II.

As análises estatísticas dos resultados revelaram que o desempenho do SA_MOHS foi superior ao do NSGA-II em todas as funções e métricas avaliadas, comprovando que os aprimoramentos sugeridos, apesar de serem simples, mostraram-se bastante efetivos. Ressaltamos, no entanto, que nossas conclusões limitam-se às condições dos experimentos realizados, descritas no corpo do trabalho.

Nossa pesquisa atual envolve o desenvolvimento de uma metaheurística multiobjetivo híbrida, baseada nos métodos *Harmony Search* e *Differential Evolution*.

Referências

Chen, M. e Lu, Y. (2008), A novel elitist multiobjective optimization algorithm: Multiobjective extremal optimization. European Journal of Operational Research, 188, 637–651.

Coello Coello, C. A. A Short Tutorial on Evolutionary Multiobjective Optimization, in Zitzler, E. *et al* (Eds.). Evolutionary Multi-Criterion Optimization. Springer, Berlin, 21-40, 2001.

Coello Coello, C. A., Lamont, G. B. e Van Veldhuizen, D. A. Evolutionary Algorithms for solving Multi-Objective Problems, 2nd ed, Springer, New York, 2007.

Deb, K. Multi-Objective Optimization using Evolutionary Algorithms, 1st ed, John Wiley & Sons, Chichester, 2001.

Deb, K. *et al* (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions On Evolutionary Computation, 6, 182-197.

Deb, K. e Goyal, M. (1996), A Combined Genetic Adaptive Search (GeneAS) for Engineering Design. Computer Science and Informatics, 26, 30-45.

Fonseca, C. M. e Fleming, P. J. (1993), Genetic Algorithms for Multiobjective Optimization: Formulation, Discution and Generalization. Genetic Algorithms: Proceedings of the Fifth International Conference. San Mateo, CA.

Geem, Z. W., Kim, J. H. e Loganathan, G. A. (2001), New Heuristic Optimization Algorithm: Harmony Search. Simulation, 76, 60-68.

Goldberg, D. E. Genetic Algorithms in Search, Optimization and Machine Learning, 1st ed, Addison-Wesley, Boston, 1989.

Konak, A., Coit, D. W. e Smith, A. E. (2006), Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering and System Safety, 91, 992–1007.

R Development Core Team. R: A Language and Environment for Statistical Computing, Vienna, 2011. ISSN 3-900051-07-0. Disponivel em: <http://www.R-project.org/>.

Ricart, J. et al (2011), Multiobjective Harmony Search Algorithm Proposals. Electronic Notes in Theoretical Computer Science, 281, 51-67.

Trautmann, H., Steuer, D. e Mersmann, O. (2010), MCO: Multi criteria optimization algorithms and related functions. R package version 1.0.9. Disponivel em: <http://CRAN.R-project.org/package=mco>.

Wonnacott, T. H. e Wonnacott, R. J. Introductory Statistics, 5th ed, John Wiley & Sons, New York, 1990.

Yang, X. S. Nature-Inspired Metaheuristic Algorithms. 1st ed. Luniver Press, Frome, 2008.

Zitzler, E., Deb, K. e Thiele, L. (2000), Comparison of Multiobjective Evolutionary Algorithms: Empirical results. Evolutionary Computation, 8, 173-195.

Zitzler, E., Laumanns, M. e Bleuler, S. A Tutorial on Evolutionary Multiobjective, in Gandibleux, X., Sevaux, M. e Sörensen, K. (Eds.). Metaheuristics for Multiobjective Optimisation. Springer, Berlin, 535, 3-38, 2004.

Zitzler, E., Laumanns, M. e Thiele, L. (2001), SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Swiss Federal Institute Techonology, Zurich, p. 21.