Evolução Diferencial Multiobjetivo Híbrido com K-Means e NSGA II: Uma Análise Comparativa frente ao NSGA III

Ciniro Ap. Leite Nametala[1], Gisele Lobo Pappa[2] e Eduardo Gontijo Carrano[3]

[1]Departamento de Engenharia e Computação
Instituto Federal de Minas Gerais, Bambuí, MG, Brasil
[2]Departamento de Ciência da Computação
[3]Departamento de Engenharia Elétrica
Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil
ciniro@ifmg.edu.br, glpappa@dcc.ufmg.br, carrano@cpdee.ufmg.br

Resumo Na atualidade um dos algoritmos mais eficientes para a busca de soluções globais em problemas de otimização multiobjetivos é o chamado NSGA III (*Non-Dominated Sorting Genetic Algorithm III*). Este vem sendo aplicado nos últimos anos a diversas classes de problemas e apresentando bons resultados, fato que popularizou a sua utilização tanto na indústria como em ambientes de pesquisa. Pode-se observar na mais recente literatura da área que diversos algoritmos vêm sendo desenvolvidos a fim de gerar resultados melhores ou equivalentes aos obtidos pelo NSGA III. Este trabalho propõe um algoritmo evolucionário multiobjetivo híbrido baseado no Evolução Diferencial, no *K-means* e no NSGA II. Os resultados da aplicação deste nos *benchmarks* DTLZ1 e DTLZ2 para 3 e 5 objetivos mostraram que a hibridização de três algoritmos clássicos diferentes pode gerar resultados equivalentes aos obtidos pelo NSGA III. As métricas e condições de experimentação desta análise são as mesmas usadas pelos autores do NSGA III para demonstrar sua qualidade.

Keywords: Otimização Multiobjetivos, Computação Evolucionária, NSGA III, Clusterização, Evolução Diferencial

1 Introdução

O uso da abordagem evolucionária em algoritmos para problemas de otimização multiobjetivos é referenciada na literatura como MOEA, sigla do inglês para

Multiobjective Optimization Evolutionary Algorithms. MOEA têm sido utilizados para resolver diferentes instâncias de problemas com eficiência. Trabalhos de destaque como os de Coello [1] e Deb [2] mostram algumas dessas aplicações. Outros, como em Takahashi *et. al.* [3] descrevem como MOEA são capazes de tratar problemas com funções multimodais em múltiplas bacias de atração com diferentes mínimos locais. Vantagens como a redução de complexidade e a capacidade de adaptação são também apresentadas por diversos trabalhos da área. Muitos MOEA ganharam notoriedade por resolver problemas de otimização simulando situações recorrentes da natureza, fato que os classifica como algoritmos bioinspirados, contudo, ao longo dos últimos anos, outras implementações se destacaram mesmo não sendo bioinspiradas. Pode-se citar dentre estas, conforme Takahashi *et. al.* [3], o Recozimento Simulado, Estimação de Distribuição, Busca Tabu, Busca Gulosa Adaptativa (GRASP) e, no interesse deste estudo, o algoritmo de Evolução Diferencial (ED).

O ED foi proposto pela primeira vez na publicação de Storn e Price [4], entretanto ganhou notoriedade após obter destaque em competições da área em 1996 e 1997. Desde então modificações já foram apresentadas para o mesmo, em especial, as que possuem foco em problemas multiobjetivos (MOED). Nesse contexto este trabalho propõe o desenvolvimento de um MOED híbrido[1]. O mesmo foi alterado para que na etapa de seleção, os indivíduos não fossem aleatoriamente selecionados (como na versão original) e sim, estrategicamente escolhidos por meio da aplicação do algoritmo de clusterização *K Means* [5]. Outras modificações foram realizadas de forma inspirada no NSGA II [6], principalmente para a parte de competição e elitismo entre indivíduos. Os resultados obtidos com a aplicação nos *benchmarks* DTLZ1 e DTLZ2 com 3 e 5 objetivos [7] foram comparados ao NSGA III [8], um dos algoritmos mais eficiente e com mesmo propósito publicado até o momento. Observou-se que o MOED híbrido pode gerar soluções próximas as do NSGA III e em alguns cenários, inclusive, apresentando mesma qualidade de convergência.

Este artigo está dividido em mais quatro partes além desta introdução. A seção 2 trata da caracterização do problema descrevendo os *benchmarks* onde o algoritmo foi aplicado. A seção 3 expõe uma conceituação de cada um dos algoritmos envolvidos e o algoritmo híbrido proposto. Na seção 4 são apresentados e comentados os resultados bem como as métricas utilizadas e, por fim, na seção 5 são feitas as conclusões.

[1] O código-fonte do algoritmo híbrido, em linguagem MATLAB®, pode ser obtido em http://github.com/ciniro.

2 Caracterização do Problema

Métricas escalonáveis para avaliação de MOEA são importantes pois permitem testar diferentes algoritmos sob a mesma perspectiva. Qualquer problema de teste escalonável deve permitir a realização de avaliações para M objetivos sem que seja modificada a representação matemática do problema. Os *set* de problemas de teste DTLZ propostos por Deb *et. al.* [7] dispõe de nove *benchmarks*, dentre estes são comentados brevemente a seguir os dois selecionados para este estudo. Para maiores detalhes, consultar a referência original.

2.1 DTLZ1

Problema M objetivos com uma fronteira Pareto-ótima linear. Com base na sua formulação, assumindo-se para qualquer função $g \geq 0$, a função $g(\boldsymbol{x})$ irá requerer $\mid \mathsf{x}_M \mid = k$ variáveis. Neste modelo a definição de $g(\mathsf{x}_M)$ pode variar. A forma utilizada neste trabalho é a mesma sugerida por Deb *et. al.* [7] e pode ser vista na Eq.(1):

$$g(\mathsf{x}_M) = 100\left[\mid \mathsf{x}_M \mid + \sum\nolimits_{x_i \in \mathsf{x}_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right] \tag{1}$$

Buscando-se convergir para um hiperplano, a solução $\mathsf{x}_M = 0$ corresponde a fronteira Pareto-ótima sendo que, neste caso, $\sum_{m=1}^{M} f_m = 0.5$. Para $n = M + k - 1$ o espaço de busca conterá $(11^k - 1)$ fronteiras Pareto-ótimas locais.

2.2 DTLZ2

É baseado na representação genérica da esfera. Como em DTLZ1 a função $g(\mathsf{x}_M)$ também é definida com $\mid \mathsf{x}_M \mid = k$ variáveis. Para DTLZ2 foi utilizada a formulação da Eq.(2).

$$g(\mathsf{x}_M) = \sum\nolimits_{x_i \in \mathsf{x}_M} (x_i - 0.5)^2 \tag{2}$$

Os valores referentes a x_i neste problema estão contidos no intervalo $[0, 1]$. A solução ótima em $g(\mathsf{x})$ é definida quando $x_M^* = 0.5$ e satisfeita a condição observada na Eq.(3).

$$\sum_{i=1}^{M} (f_i^*)^2 = 1 \tag{3}$$

3 Metodologia

Nas subseções a seguir são apresentados em suas formas clássicas os algoritmos ED, *K means* e NSGA II. Na sequencia é feito um detalhamento das parametrizações, etapas e do pseudocódigo referente algoritmo híbrido.

3.1 Algoritmos base

O ED tem início com uma população de N indivíduos onde cada indivíduo é composto de D variáveis, geram-se então aleatoriamente soluções que irão compor a primeira população representada de acordo com a Eq. 4, onde $i = \{1, 2, \ldots, N\}$ corresponde ao índice do indivíduo na população e $t = \{1, 2, \ldots, t_{max}\}$ indica a geração.

$$X_i^t = \left[x_{1,i}^t, x_{2,i}^t, \ldots, x_{D,i}^t\right] \tag{4}$$

Em $t = 0$ a geração da população inicial é feita com base em uma distribuição uniforme $U(0, 1)$ para cada variável D que compõe um dado indivíduo. Obedecendo-se limites mínimos e máximos pré-estabelecidos este procedimento ocorre como exibido na Eq. 5.

$$x_{j,i} = x_{j,min} + rand_{i,j}[0, 1] \times (x_{j,max} - x_{j,min}) \tag{5}$$

Considerando a seleção aleatória de três indivíduos x_{r1}, x_{r2} e x_{r3} na população corrente sendo que $r1 \neq r2 \neq r3$, o ED em sua forma clássica [9] atua conforme a Eq. 6 quando instanciado o operador de mutação. Nesta etapa, F diz respeito a um fator de mutação aplicado a diferença entre dois dos indivíduos selecionados. Essa diferença é adicionada ao terceiro indivíduo. O resultado é o chamado vetor *donor* (V_i).

$$\boldsymbol{V_i} = \boldsymbol{x_{r3}} + F \cdot (\boldsymbol{x_{r2}} - \boldsymbol{x_{r1}}) \tag{6}$$

Após a etapa de mutação é selecionado aleatoriamente um último indivíduo, este é chamado vetor *target*. Os vetores *target* e *donor* são então submetidos ao operador de cruzamento que é apresentado na Eq. 7. Este efetuará a troca de variáveis entre os indivíduos com base numa taxa de cruzamento C_r ou em um número aleatório j_{rand} gerado entre $[1, D]$. O uso do valor de j_{rand} é necessário

para que pelo menos uma variável do vetor *donor* seja repassada a U_i.

$$U_{i,j} = \begin{cases} V_{i,j}, & se \quad U(0,1) \leq C_r \quad ou \quad j = j_{rand} \\ X_{i,j}, & caso \quad contrário \end{cases} \tag{7}$$

O resultado do processo de seleção, mutação e cruzamento é chamado de vetor *trial* (U_i). Este competirá com o vetor *target* (X_i) a fim de determinar qual destes irá ser transferido para a população $t + 1$.

Já o *K means* é um algoritmo de agrupamento não hierárquico. Este minimiza a distância de um dado ponto amostral para um dado centroide. A minimização ocorre por meio de iterações que calculam o local no espaço onde exista a mínima variância entre os elementos de um mesmo grupo. A forma de clusterização proposta por meio desse algoritmo foi publicada por Hartigan [5] em 1979. Neste trabalho optou-se pelo *K means* frente a outros algoritmos de clusterização devido a sua natural aplicabilidade no ED que, necessitando de uma quantidade fixa de indivíduos para operar, dispensa abordagens para cálculo do número ideal de *clusters*.

Por fim, o NSGA II [6] é uma evolução do algoritmo NSGA proposto pelo mesmo autor. Seu principal mecanismo é o conceito de dominância entre indivíduos. A dominância busca dividir as soluções em fronteiras distintas privilegiando àquelas que estão mais próximas da condição de otimalidade. As melhores soluções convergem para uma fronteira Pareto-ótima e, neste contexto, são consideradas *não dominadas*. Outra característica importante deste algoritmo é a implementação do conceito de distância de multidão [6]. A distância de multidão corresponde a uma análise da vizinhança existente no entorno de uma dada solução no espaço de busca. Indivíduos com poucos vizinhos são privilegiados, pois indicam soluções dispostas longe de mínimos locais evidenciando assim pontos alternativos do espaço de soluções.

3.2 Algoritmo híbrido proposto

O algoritmo híbrido foi desenvolvido com base nos passos clássicos de um algoritmo evolucionário comum sendo estes: Parametrização, geração e avaliação da população inicial, aplicação dos operadores genéticos (seleção, mutação, cruzamento e competição), geração da nova população e, por fim, avaliação do critério de parada. O pseudocódigo 1 dispõe a implementação completa que deve ser uti-

lizada pelo leitor de forma a acompanhar as subseções a seguir. Estas mostram as particularidades de cada etapa.

Inicialização. O algoritmo tem início com a atribuição dos valores desejados aos parâmetros iniciais. Na sequência é gerada uma população inicial por meio do método $InicializaPopulacaoED()$. Neste ponto cada indivíduo é gerado como feito no ED clássico, ou seja, conforme Eq. 5 da seção 3.1. O método $AvaliaPopulacao()$, que vem a seguir, calcula a *fitness* dos indivíduos (conforme mostrado na seção 2) em cada um dos objetivos no problema de interesse. A quantidade de objetivos é determinada pela variável $N_{objetivos}$ e o problema a ser analisado pela variável *problema*. Por fim, ainda na inicialização, o parâmetro K é setado com a finalidade de determinar a quantidade de indivíduos necessários quando da aplicação do K *means*, no caso, sete (3 pares competidores, além do vetor *target* - detalhes na subseção sobre mutação).

Evolução. As iterações que garantem a evolução são interrompidas com base no número de chamadas a função objetivo. Este valor é determinado pela variável $N_{calculos}$. Enquanto este valor não for excedido, independente da geração, o processo evolucionário irá continuar. No *loop* das gerações, a cada iteração, é atualizado o valor do fator de mutação F (f_t) conforme uma função de decaimento com curvatura β setada em 170, valor definido após avaliações. A função em questão pode ser observada nas equações 8 , 9 , 10 e 11, onde t é a geração corrente e t_{total} o número máximo de gerações. A variação dinâmica ocorre de $f_{max} = 1.2$ (momento de favorecimento máximo a exploração) a $f_{min} = 0.2$ (momento de favorecimento máximo a explotação). Este é um ponto que diferencia este algoritmo de outros MOED comumente vistos na literatura, pois em geral é comum observa-se estratégias em que o fator F é constante (como no algoritmo original).

$$f_{inf} = -(e^{t_{total}/\beta})^2 \times 0.01 + 1 \tag{8}$$

$$f_{sup} = -(e^{1/\beta})^2 \times 0.01 + 1 \tag{9}$$

$$f_{ref} = -(e^{t/\beta})^2 \times 0.01 + 1 \tag{10}$$

$$f_t = \frac{(f_{ref} - f_{inf})}{(f_{sup} - f_{inf})} \times (f_{max} - f_{min}) + f_{min} \tag{11}$$

Mutação. O operador de mutação irá ser utilizado conforme probabilidade definida na variável $Taxa_{mutacao}$. Neste estudo utilizou-se 60%. Caso o operador

seja acionado é chamado o método $SelecaoKMEANS()$ que divide a população corrente em sete *clusters*. O *K means* foi aplicado com distância *euclidiana* e tipo de aglomeração por *média aritmética*. De cada um dos *clusters* é retirado um indivíduo aleatório. Um destes indivíduos irá ser considerado como vetor *target* (V_{target}), enquanto que os outros 6 irão formar 3 pares distintos. Em cada par é aplicado o método $CompeticaoNSGAII()$. O método $CompeticaoNSGAII()$ recebe dois indivíduos e os compara, inicialmente, utilizando *dominância*. Caso o resultado seja um empate é aplicado então nova comparação, desta vez baseada em *distância de multidão*. Os indivíduos vencedores de cada par irão ser submetidos ao processo de mutação clássico do ED (Conforme apresentado na Eq. 6) para gerar o vetor *donor* (método $MutacaoED()$). No caso do operador de mutação não ser acionado, são selecionados aleatoriamente dois indivíduos na população. Estes serão submetidos como *target* e *donor* à próxima etapa, o operador de cruzamento.

Cruzamento. A variável $Taxa_{cruzamento}$ diz respeito a probabilidade de ocorrência deste operador. Neste estudo a mesma foi fixada em 90%. No caso da operação não ser acionada, o vetor *target* é simplesmente incluído na nova população ($Populacao_{nova}$) como um novo indivíduo. Porém, quando o operador é acionado gera-se o vetor *trial* por meio do método $CruzamentoED()$. Este método promove um cruzamento ponto a ponto baseado no valor da variável $Taxa_{polarizacao}$. A taxa de polarização é um outro diferencial deste algoritmo híbrido e diz respeito a possibilidade de privilegiar, ou não, o material genético do vetor *donor* em detrimento do *target*. Utilizou-se uma taxa de polarização fixa de 90%. Após o cruzamento é gerado então o vetor *trial* (V_{trial}) que na sequencia é avaliado pelo método $AvaliaIndividuo()$. O $AvaliaIndividuo()$ usa os mesmos critérios do método $AvaliaPopulacao()$ explicado anteriormente. Por fim, os vetores *trial* e *target* competem entre si. O melhor entre os dois será inserido na nova população.

Melhor solução. Ao ser satisfeito o critério de parada, os indivíduos contidos na última população (fronteira obtida) são comparados à média dos indivíduos contidos na fronteira Pareto-ótima. O indivíduo com maior proximidade é retornado como melhor solução ($melhor_{solucao}$). Para este trabalho em específico foram implementados, também na última etapa do algoritmo, diversas rotinas que avaliam a sua qualidade. As métricas utilizadas e os resultados obtidos são detalhados na seção a seguir.

Algoritmo 1: Algoritmo Híbrido Proposto

Entrada: $Taxa_{mutacao}, Taxa_{cruzamento}, Taxa_{polarizacao}, N_{calculos}$, $\text{Tam}_{populacao}, problema, N_{objetivos}$

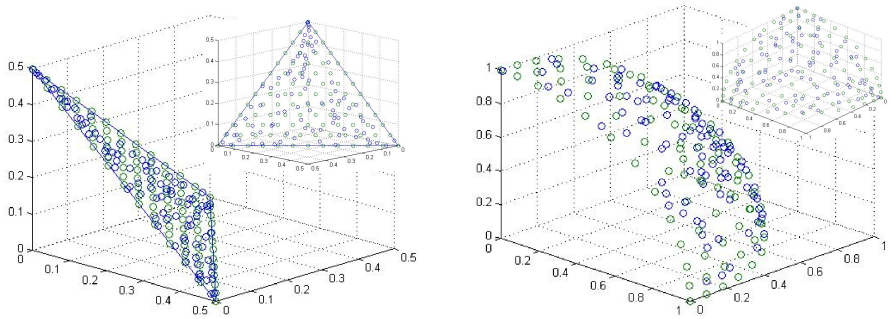
Saída: $melhor_{solucao}$

início
  $Populacao_{corrente} \leftarrow InicializaPopulacaoED(Tam_{populacao})$
  $AvaliaPopulacao(Populacao_{corrente}, problema, N_{objetivos})$
  $k \leftarrow 7$
  enquanto $cont_{calc} \leq N_{calculos}$ **faça**
    $F \leftarrow AtualizaF(cont_{calc}, N_{calculos}, Tam_{populacao})$
    $Populacao_{nova} \leftarrow InicializaPopulacaoVazia(Tam_{populacao})$
    enquanto $cont_{pop} \leq Tam_{populacao}$ **faça**
      se $Taxa_{mutacao} \leq rand(0,1)$ **então**
        $individuos \leftarrow SelecaoKMEANS(Populacao_{corrente}, k)$
        $x_{r1} \leftarrow CompeticaoNSGAII(individuos[1], individuos[2])$
        $x_{r2} \leftarrow CompeticaoNSGAII(individuos[3], individuos[4])$
        $x_{r3} \leftarrow CompeticaoNSGAII(individuos[5], individuos[6])$
        $V_{target} \leftarrow individuos[7]$
        $V_{donor} \leftarrow MutacaoED(x_{r1}, x_{r2}, x_{r3}, F)$
      senão
        $individuos \leftarrow SelecionaIndividuosAleatorios(Populacao_{corrente}, 2)$
        $V_{target} \leftarrow individuos[1]$
        $V_{donor} \leftarrow individuos[2]$
      fim
      se $Taxa_{cruzamento} \leq rand(0,1)$ **então**
        $V_{trial} \leftarrow CruzamentoED(V_{donor}, V_{target}, Taxa_{polarizacao})$
        $AvaliaIndividuo(V_{trial}, problema, N_{objetivos})$
        $Populacao_{nova}[cont_{pop}] \leftarrow CompeticaoNSGAII(V_{target}, V_{trial})$
      senão
        $Populacao_{nova}[cont_{pop}] \leftarrow V_{target}$
      fim
      $cont_{pop} \leftarrow cont_{pop} + 1$
    fim
    $Populacao_{corrente} \leftarrow Populacao_{nova}$
    $AvaliaPopulacao(Populacao_{corrente}, problema, N_{objetivos})$
    $cont_{calc} \leftarrow cont_{calc} + Tam_{populacao}$
  fim
  $melhor_{solucao} \leftarrow BuscaMelhorIndividuo(Populacao_{corrente})$
fim
retorna $melhor_{solucao}$

4 Resultados

Com o objetivo de manter as mesmas condições de experimentação utilizadas por Deb e Jain [10] no artigo em que estes reportam a aplicação do algoritmo NSGA III, os resultados apresentados nesta seção tratam de 20 execuções consecutivas do algoritmo híbrido. O melhor caso gerado nesta bateria de testes foi determinado por meio da métrica *Inverted Generational Distance* (IGD) sugerida por Sierra e Coello [11]. Esta fornece a distância média entre as soluções contidas na população corrente e o elemento mais próximo da fronteira Pareto-ótima.

Na primeira análise pode-se avaliar a fronteira gerada e a distribuição das soluções. Para isso foi utilizado como referência a fronteira Pareto-ótima de cada problema (DTLZ1 e DTLZ2) em cada configuração (3 e 5 objetivos). Vê-se que tanto na Fig. 1(a) quanto na Fig. 1(b), para 3 objetivos, em ambos os problemas a aproximação da solução gerada (pontos azuis) com a fronteira Pareto-ótima (pontos verdes) foi satisfatória. Isso pode ser constatado, pois os indivíduos estão bem distribuídos ao longo de todo o espaço e não existem ocorrências de *outliers* que denotem qualquer distância visualmente observável para o hiperplano. Os intervalos de [0.5,0.5,0.5] em DTLZ1 e [1,1,1] em DTLZ2, que definem em cada problema os limites da fronteira ótima, em nenhuma das soluções geradas foi violado. Todas os indivíduos, neste sentido, podem ser considerados portanto candidatos a solução ótima.



(a) DTLZ1 (b) DTLZ2

Figura 1. Fronteiras Pareto obtidas para 3 objetivos

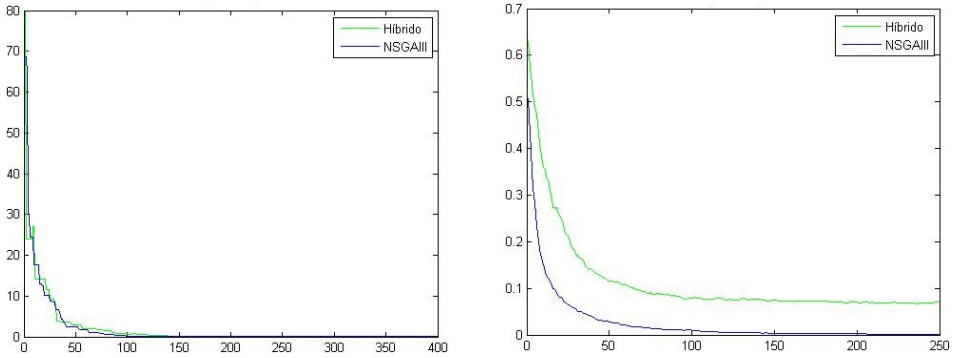
Quando comparando-se o algoritmo híbrido e o NSGA III por meio da análise gráfica da convergência ao longo das gerações e, em todas as 20 execuções, vê-se resultados de pior IGD, melhor IGD e IGD médio. Na Tab. 1 em cada cenário

Tabela 1. Comparativo entre os algoritmos para melhor IGD, IGD médio e pior IGD.

Problema	**Número de Objetivos**	**Máximo de Gerações**	**NSGA III**	**Algoritmo Híbrido**
DTLZ1	3	400	4.888×10^{-4} 1.308×10^{-3} 4.888×10^{-3}	2.490×10^{-2} 6.855×10^{-2} 2.936×10^{-1}
	5	600	5.116×10^{-4} 9.799×10^{-4} 1.979×10^{-3}	6.287×10^{-2} 6.647×10^{-2} 6.762×10^{-2}
DTLZ2	3	250	1.262×10^{-3} 1.357×10^{-3} 2.114×10^{-3}	6.769×10^{-2} 7.265×10^{-2} 7.806×10^{-2}
	5	350	4.254×10^{-3} 4.982×10^{-3} 5.862×10^{-3}	2.026×10^{-1} 2.107×10^{-1} 2.248×10^{-1}

de teste, nota-se que, apesar da boa aproximação, neste critério o NSGA III apresentou melhores resultados. Os valores mais próximos a fronteira gerada por ele correspondem a diferenças médias da ordem de 10^{-3}, ao passo que no algoritmo híbrido observou-se melhores diferenças na ordem de 10^{-2} e piores na ordem de 10^{-1}. A boa qualidade dos resultados em todos os cenários testados mostra que o algoritmo híbrido, mesmo neste caso não vencendo o NSGA III, apresenta-se como uma alternativa válida, especialmente quando adotada uma tolerância mínima para a solução ótima no intervalo de 10^{-2} a 10^{-1}, valores que para problemas de otimização desta natureza, em geral, podem ser considerados de utilidade.

Quanto a convergência não observou-se maiores diferenças entre o NSGA III e o algoritmo híbrido quando submetidos a DTLZ1 seja com 3 ou 5 objetivos. Neste *benchmark* pode-se notar equivalência entre ambos, fato que pode ser visualizado na Fig. 2(a) que mostra, por exemplo, a evolução baseada em IGD médio ao longo de 400 gerações. Já em DTLZ2 com 3 objetivos e 250 gerações pode-se ver, conforme na Fig. 2(b), uma forma na curva de convergência do híbrido apenas semelhante a do NSGA III. Isso sugere que ambos convergem com a mesma velocidade (por volta da 100º geração). No entanto, os resultados finais menos próximos da solução ótima gerados pelo híbrido denotam uma discrepância quanto ao IGD médio entre os dois, vencendo neste caso o NSGA III. Como em DTLZ1, em DTLZ2 a convergência para 5 objetivos apresenta comportamento semelhante ao obtido no cenário com 3 objetivos.



(a) DTLZ1 (IGD-m x Gerações) (b) DTLZ2 (IGD-m x Gerações)

Figura 2. Convergência das populações para 3 objetivos

5 Conclusões

O algoritmo NSGA III é citado na mais recente literatura da área como um dos algoritmos mais eficientes na resolução dos problemas do conjunto DTLZ. Diversos trabalhos como Yuan *et. al.* [12] e Bandyopadhyay [13] comparam sua aplicabilidade frente a outros algoritmos multiobjetivos diferentes. O entendimento, implementação, adaptação e uso destes algoritmos exige estudo e dedicação. Isso ocorre, pois além da complexidade, estes apresentam também estratégias que só foram divulgadas recentemente. Diante destes pontos, o principal objetivo deste estudo foi o de privilegiar estratégias clássicas, eficazes e já difundidas. Para tanto, ao contrário de algoritmos mais recentes, pode-se citar revisões de literatura como em Hwang [14] que demonstram grande esforço de pesquisa já empreendido em algoritmos como o *K means* e outros MOEA como o NSGA II e o ED. Espera-se portanto que, com a demonstração destes resultados, seja disponibilizada uma alternativa equivalente ao NSGA III em muitos aspectos. Uma alternativa que apresenta assim, por principal vantagem, o uso de algoritmos conhecidos, fato que, em muitas situações, pode gerar facilidades tanto na implementação/aproveitamento de código quanto na adaptação do mesmo a problemas diferentes dos aqui testados.

Pretende-se em trabalhos futuros avaliar o impacto da utilização de diferentes tipos de distância e tipos de aglomeração no uso do *K means*. Outra proposta de continuidade é ampliar a quantidade de problemas de teste.

Referências

1. C. A. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36 (2006)
2. Kalyanmoy Deb. *Multiobjective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons (2001)
3. Ricardo H. C. Takahashi, Antônio Gaspar-Cunha, and Carlos M. Fonseca. *Manual da Computação Evolutiva e Metaheurística*. Editora UFMG (2013)
4. Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report TR-95-012, Internacional Computer Science Institute (ICSI), University of California, Berkeley* (1995)
5. Hartigan, J. A., Clustering Algorithms. Wiley (1979)
6. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation*, IEEE Transactions on, 6(2):182–197 (2002)
7. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns and Eckart Zitzler. *Scalable test problems for evolutionary multiobjective optimization*. Springer (2005)
8. Himanshu Jain and Kalyanmoy Deb. An improved adaptive approach for elitism nondominated sorting genetic algorithm for many-objective optimization. In: *Evolutionary Multi-Criterion Optimization*, pages 307–321. Springer (2013)
9. Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359 (1997).
10. Kaushik Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *Evolutionary Computation*, IEEE Transactions on, 18(4):577–601 (2014)
11. Sierra, M.R. e Coello, C.A.C., Improving pso-based multi-objective optimization using crowding, mutation and mutation. In: *Evolutionary multi criterion optimization*. Springer, 505–519 (2005)
12. Yuan, Y., Xu, H. e Wang, B., An improved nsga-iii procedure for evolutionary many-objective optimization. In: *Proceedings of the 2014 conference on Genetic and evolutionary computation*. ACM, 661–668 (2014)
13. Bandyopadhyay, S. e Mukherjee, A., An algorithm for many-objective optimization with reduced objective computations: A study in differential evolution. *Evolutionary Computation*, IEEE Transactions on, Vol. 19, No. 3, 400–413 (2015)
14. Hwang, C.L. e Masud, A.S.M., Multiple objective decision making—methods and applications: a state-of-the-art survey, Vol. 164. *Springer Science & Business Media* (2012)