Algoritmo cultural com busca local avaliado através de testes estatísticos não paramétricos

Carlos A. O. de Freitas[1], Roberto C.L. Oliveira[2], Deam J.A. Silva[3], Jandecy C. Leite[4], Jorge L.M. Rodriguez[4]

[1]PPGEE – Universidade Federal do Pará (UFPA)
Belém – PA – Brasil
[2]Laboratório de Computação Bio-Inspirada – Universidade Federal do Pará (UFPA)
Belém – PA - Brasil
[3]IEG – Universidade Federal do Oeste do Pará (UFOPA)
Santarém – PA – Brasil
[4]Instituto de Tecnologia Galileo da Amazônia (ITEGAM)
Manaus – AM – Brasil

carlos.freitas.br@ieee.org, limao@ufpa.br,
deam.silva@ufopa.edu.br, jandecy.cabral@itegam.org.br,
jorgemoyar@gmail.com

Resumo. Este trabalho tem como objetivo analisar o desempenho do clássico algoritmo cultural (CA) com uma nova proposta CA além de duas técnicas de pesquisa locais (Simulated Annealing - SA e Busca Tabu - BT). Para diversificar os testes, no AC com SA houve variação da energia do parâmetro, e no AC com BT, houve variação no tamanho da lista de tabu. Os algoritmos foram submetidos a dois cenários (cenário 1 - Funções básicas, cenário 2 - Funções híbridas). O algoritmo proposto difere de outros encontrados na literatura, pelo processo de alimentação do conhecimento topográfico que orienta a pesquisa. Já que é alimentado pelos espaços onde as buscas locais tiveram o melhor desempenho formando uma área de resultados promissora, isso justifica o contributo da pesquisa. A análise foi realizada utilizando os testes Friedman, Friedman Aligned e Quades, que servem para comparar o comportamento de um conjunto de algoritmos de uma só vez.

Palavras chaves: Busca Tabu, Algoritmo Cultural, Medida de Desempenho, *Simulated Annealing*.

1 Introdução

Já há algum tempo a ciência procura modelar a evolução natural dos seres vivos em sistemas computacionais [1, 2]. Do ponto de vista da engenharia, esses modelos serviram como base para o desenvolvimento de meta-heurísticas para solução de problemas, basicamente em otimização de sistemas [3-5]. O avanço das pesquisas demonstrou que meta-heurísticas dotadas de mecanismos distintos de funcionamento podem

ser mais adequadas para problemas com determinadas estruturas, e outras meta-heurísticas podem funcionar melhor em outras classes de problemas [6]. Isto conduziu as pesquisas para o desenvolvimento de novas meta-heurísticas que se basearam em outros processos da natureza diferentes da evolução da espécie. Devido às novas abordagens, nas quais percebe-se a ocorrência do aumento no conhecimento dos mecanismos que fundamentam os algoritmos da computação evolutiva, nota-se que os novos Algoritmos Evolutivos (AEs) estão se afastando da estrita inspiração biológica. Os novos AEs tendem a aprofundar a tendência de incorporação de operações e mecanismos que não sejam bio-inspirados, mas sim inspirados em argumentos matemáticos ou computacionais [6]. Também temos algoritmos inspirados na adaptação e evolução cultural dos indivíduos em uma comunidade, chamados de Algoritmos Culturais (ACs). Estes geram ou alteram seus conhecimentos em função do relacionamento entre os indivíduos da comunidade. Os Algoritmos Culturais (ACs ou AC) foram propostos por [7]. Devido suas características de paralelismo implícito e busca aleatória os ACs são utilizados na solução de problemas tradicionais de otimização complexos [8]. Técnicas que empregam meta-heurísticas (AGs, ACs, etc.) como busca global e heurísticas de busca local (*hill climbing*, BT, SA, etc.), são comumente referidos como Algoritmos Meméticos (AMs) [9], ou algoritmos híbridos. Normalmente os AMs não só podem apresentar uma boa capacidade exploratória, similar ao que faz um algoritmo com base populacional de busca global, mas também proporciona um bom desempenho de intensificação durante a busca, semelhante ao que faz um algoritmo de busca local. A hibridização dos ACs com o mecanismo de busca local para exploração extensiva nas soluções geradas pelos ACs, pode melhorar em muito o desempenho deste hibrido em relação aos algoritmos na sua forma clássica. As comparações de AEs geralmente tendem para a análise de seus resultados após diversas execuções destes na tentativa de solucionar várias funções de benchmarks. Essas avaliações são realizadas por meio de hipóteses estatísticas [10, 11]. Pode-se dizer que se trata de analises baseadas em resultados [12], ou seja, é uma avaliação do desempenho do algoritmo para certas funções de benchmark. Porém, a dificuldade está na comparação de diversos algoritmos, pois geralmente esta comparação é realizada em pares de algoritmos[13] e aumenta com o número de algoritmos a serem avaliados, além de aumentar a probabilidade de fazer um erro [14]. O interesse na análise estatística não paramétrica tem crescido recentemente no campo da inteligência computacional [10], pois a mesma pode ser uma forma de comparar algoritmos evolutivos, testados para vários problemas diferentes, com alguma significância estatística. Nesta proposta apresenta-se a hibridação dos ACs com duas formas de busca local (BT e SA) são comparadas entre si e com os ACs clássicos. Estes três algoritmos (ACs puros (clássico), ACs com BT e ACs com SA) são utilizadas para encontrar o mínimo de oito funções benchmarks de variáveis reais. Os resultados são avaliados com base nos testes não paramétricos: Friedman, Friedman Aligned e Quade.

O artigo proposto disponibiliza na seção 2 um conteúdo básico de itens tidos como base para desenvolvimento deste. Na seção 3 são apresentados os materiais e métodos com os cenários de testes utilizados neste trabalho. Já na seção 4, são apresentados os resultados das simulações realizadas. Finalmente, a seção 5 conclui o trabalho com as observações e comentários sobre as simulações realizadas.

2 Meta-Heurísticas para Otimização

Nesta seção serão apresentadas as meta-heurísticas de busca global 'Algoritmo Cultural' e as duas meta-heurísticas utilizadas para busca local '*Simulated Annealing'* e 'Busca Tabu'. A combinação de um algoritmo de busca global com um algoritmo de busca local compõe a base da Computação Memética [15].

2.1 Algoritmos Culturais

Os ACs são usados para modelar a evolução do componente cultural em um sistema evolutivo computacional ao longo do tempo, uma vez que acumula experiência na resolução em um conjunto de dados na resolução de problemas [8]. A evolução cultural permite que as sociedades envolvam ou adaptem seu meio ambiente a taxas que excedem a evolução biológica, que se baseia apenas na herança genética [7].

Os ACs são formados basicamente de um espaço populacional, um espaço de crenças, protocolos de comunicação (Funções de Aceitação e Influência) entre os dois espaços e algumas funções auxiliares: Inicialização, Seleção, Atualização e Avaliação. A estrutura dos ACs é mostrada na Fig. 1, o seu pseudocódigo é apresentado na Fig. 2.

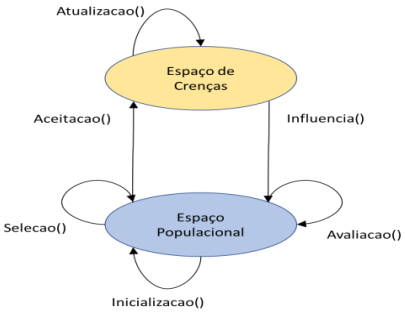

Fig. 1. Estrutura do Algoritmo Cultural.
Fonte: Adaptado de [16].

```
01: INICIO
02: t=0;
03: Inicializa B(t) e P(t)
04: REPITA
05: Avalia P(t) usando Avaliacao()
06: Atualiza B(t) usando Aceitacao()
07: Gera P(t) usando influencia()
08: t=t+1
09: Seleciona P(t) de P(t-1)
10: ENQUANTO (condição de término encontrada)
11: FIM
```

Fig. 2. Pseudocódigo do Algoritmo Cultural.
Fonte: Adaptado de [16].

Os espaços citados são descritos a seguir:

Espaço Populacional: conjunto de soluções que pode ser modelado utilizando qualquer técnica que faça uso de uma população de indivíduos;

O Espaço de Crenças (Mapa do Grupo): é o local onde ocorre o armazenamento e representação do conhecimento (experiência ou mapas individuais) adquirido ao longo do processo evolutivo. As fontes de conhecimento são cinco conforme [16], estas são úteis na tomada de decisões [16, 17]. Por exemplo: Conhecimento situacional possui soluções com sucesso e sem sucesso, etc.; Conhecimento normativo contém

intervalos de comportamentos aceitáveis. Conhecimento topográfico possui padrões espaciais de comportamento.

O Espaço populacional e o espaço de crença são ligados por um mecanismo (protocolo) de comunicação composto por uma função de aceitação que é usada para coletar a experiência de indivíduos da população selecionada. A outra função do protocolo de comunicação é a função de influência que pode fazer uso do conhecimento de soluções de problemas no espaço de crença para orientar a evolução de indivíduos no espaço populacional. Os ACs podem explorar tanto a microevolução quanto a macro evolução. A microevolução diz respeito à evolução que acontece no nível populacional e a macroevolução é a que ocorre sobre a cultura em si, ou seja, a evolução do espaço de crenças [18].

2.2 Simulated Annealing

O *Simulated Annealing* (SA) é uma meta-heurística inspirada no processo físico de recozimento de um sólido para obtenção de estados de baixa energia na área da física da matéria condensada [6]. O SA estabelece uma ligação entre esse tipo de comportamento termodinâmico e a busca de mínimos globais para um problema de otimização discreta.

Da mesma forma que o sólido é resfriado lentamente para garantir uma estrutura cristalina, o algoritmo resfria a solução lentamente para garantir que ela tenha a melhor função objetivo ao mesmo tempo em que permite configurações que vão de encontro ao melhor valor da função objetivo encontrado (situação correspondente a pequenos aquecimentos) [18].

A aceitação de configurações que têm tempera mais elevada, para [18] essa é uma característica importante do SA, que pode parecer pior, ou seja, permite a aceitação de uma configuração que proporciona um "pior" valor para a função objetivo, evitando assim a convergência para um local mínimo. Esta aceitação é determinada por um número aleatório sendo controlado pela expressão (1):

$$P = e^{-\Delta/T} \tag{1}$$

2.3 Busca Tabu

A busca tabu (BT) guia um procedimento heurístico de busca local pela utilização de características da solução corrente e da história da busca para explorar o espaço de soluções. [19], em vários casos, os métodos descritos proporcionam soluções muito próximas da solução ótima e estão entre os mais eficazes, senão os melhores, para enfrentar os difíceis problemas em questão. Sendo uma técnica de busca local, a BT parte de uma solução inicial e se move no espaço de soluções de uma solução para outra que esteja em sua vizinhança [6].

O uso sistemático de memória adaptativa constitui a propriedade que distingue BT de outras meta-heurísticas. A palavra “adaptativa” significa que a memória atualiza o armazenamento de elementos de soluções ou soluções completas encontradas durante a exploração dos espaços de soluções [19].

O processo de intensificação é melhorado pelo uso das estruturas de memória, chamadas de listas tabu. A cada iteração é verificado se a solução corrente já foi visitada anteriormente ou se violou alguma regra, se sim, esta solução é armazenada na lista tabu e marcada como "tabu". Este procedimento evita a chamada ciclagem, ou seja, que uma solução seja visitada novamente. Com esta estratégia de memória, o algoritmo BT pode ir além do ótimo local e acessar outras regiões do espaço de soluções [6]. Essa estratégia está fundamentada no fato de que na exploração do espaço de soluções, as soluções geradas a mais tempo possivelmente estão "distantes" da região do espaço sob análise e, como tal, não tem influência na escolha da próxima solução vizinha naquela região [6]. O tamanho da lista tabu é considerado um parâmetro crítico. Pois segundo [6], A dimensão da lista não pode ser tão pequena, sob pena de haver ciclagem; nem tão grande, para armazenar desnecessariamente soluções que não estejam ligadas à história recente da busca.

2.4 Testes de Comportamento de Algoritmos: *Friedman, Friedman Aligned e Quade*

A necessidade em definir o comportamento de algoritmos quando submetidos a problemas de diferentes naturezas, tem aberto um campo de pesquisa em procedimentos de testes [12, 20]. O teste de Friedman é um teste de comparações múltiplas que visa detectar diferenças significativas entre o comportamento de dois ou mais algoritmos [10]. O processo para realização do teste de Friedman segue os passos a seguir, segundo [10]:

1. Reunir todos os resultados de cada par de algoritmo/problema;
2. Classificar os valores de cada problema i de 1 (melhor resultado) até k (pior resultado). Note esta classificação como r_j^i $(1 \leq j \leq k)$;
3. Para cada algoritmo j, calcular a média das classificações obtidas em todos os problemas para obter a classificação final $R_j = \frac{1}{n} \sum_i r_i^j$.

Desta forma os algoritmos são classificados para cada problema separadamente. Como indicado no item 2, o algoritmo com melhor desempenho é classificado com 1, o segundo melhor com 2, etc.

A estatística de Friedman é calculada de acordo com a equação 2.

$$F_f = \frac{12n}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right], \tag{2}$$

Para o teste de Friedman alinhado, é calculado um valor de localização como o desempenho médio alcançado por todos os algoritmos em cada problema. O passo de obter a diferença entre o desempenho de um algoritmo e o valor da localização é repetida para cada combinação de algoritmos e problemas. A equação 3, mostra a definição para o cálculo estatístico da classificação alinhada de Friedman.

$$F_{AR} = \frac{(k-1)\left[\sum_{j=1}^{k} \hat{R}_j^2 - (kn^2/4)(kn+1)^2\right]}{\{[kn(kn+1)(2kn+1)]/6\} - (1/k)\sum_{i=1}^{n} \hat{R}_i^2} \quad , \tag{3}$$

O teste Quade é o terceiro teste utilizado neste trabalho. Este teste difere do de Friedman que considera a igualdade em termos de importância entre os algoritmos, leva em conta o fato que alguns problemas são mais difíceis ou que as diferenças registradas na sequência de vários algoritmos sobre eles são maiores. Portanto, os rankings calculados em cada problema podem ser dimensionados dependendo das diferenças observadas nos desempenhos dos algoritmos, obtendo, como resultado, uma análise de classificação ponderada da amostra de [10].

O teste de Quade pode ser calculado pela equação 4, levando em consideração algumas definições apresentadas em [10]. Considerando também os termos A e B, dados pelas equações 5 e 6, respectivamente.

$$F_Q = \frac{(n-1)B}{A-B}, \tag{4}$$

Os valores de A e B são apresentados abaixo:

$$A = n(n+1)(2n+1)k(k+1)(k-1)/72 \ , \tag{5}$$

$$B = \frac{1}{n}\sum_{j=1} kS_j^2 \quad , \tag{6}$$

2.5 Algoritmos Utilizados

Os ACs possuem em seu espaço populacional a população evoluída através dos AGs. Dentro da função de mutação são aplicadas as técnicas de busca local (O SA variando a energia de busca local em 5, 10 e 15. A BT com as variações do tamanho da lista tabu em 2, 4 e 6).

Este se diferencia de [17] que utiliza como técnica para definir a vizinhança o conceito de "*ball*" [18], onde é definida uma área de raio 'r' onde deverá conter as possíveis soluções. O uso das informações dos 3 (três) melhores indivíduos encontrados na busca local são utilizadas para definir uma área de bom comportamento que alimenta o conhecimento topográfico. Esta abordagem até então não tinha sido utilizada, de acordo com a pesquisa bibliográfica realizada, isso mostra a relevância da pesquisa. A figura 3 mostra o pseudocódigo do algoritmo utilizado. Sendo que, a função Algoritmo_Cultural_com_Busca_Local() é utilizada para representar o uso da SA ou do TB.

```
Algoritmo_Cultural_com_Busca_Local( )
01: INÍCIO
02: variáveis: g=0 (geração atual); guv=0 (geração do último valor ótimo),
03: dm (diferença mínima para chamar a BuscaLocal)
04: inicializar população P(t)              //população inicial aleatória
05: Inicializar Espaço de Crença EP(t)
06: avaliar população P(t)                  // calcula f(i) para cada indivíduo
07: ENQUANTO (não condição  fim) FAÇA
08: diferença=g - guv;                      // o valor de tu dependerá da evolução
09: Comunicação (P(t), EP(t));              // votação(Aceitação)
10: Atualização EP(t);                      // uso de operadores culturais
11: SE (diferenca>=dm) ENTÃO
12: Selecionar Melhor_Individuo da população P(t);
13: Melhor_Encontrado, Area_3_Melhores ←   BuscaLocal(Melhor_Individuo);
14: Comunicação (Melhor_Encontrado(t), Area_3_Melhores(t) ,EP(t));
15: Atualização EP(t);                      // uso de operadores culturais
16: FIMSE
17: Comunicação (EP(t), P(t));              // promoção (função de influência)
18: t←t+1;                                  // próxima geração
19: selecionar P(t) de P(t-1);
20: altera P(t);                            // crossover e mutação
21: avaliar P(t);                           // calcula f(i) para cada indivíduo
22: SE (diferenca>=z) ENTÃO
23: Selecionar Individuo  Aleatório da população P(t);
24: Melhor_Encontrado, Area_3_Melhores ←   BuscaLocal(Individuo_Aleatório);
25: Comunicação (Melhor  Encontrado(t), Area  3  Melhores(t) ,EP(t));
26: Atualização EP(t);                      // uso de operadores culturais
27: FIMSE
28: FIMENQUANTO
29:FIM
```

Fig. 3. Pseudocódigo do Algoritmo proposto.

3 Materiais e Métodos

Nesse trabalho, foram criados alguns cenários de simulação com o objetivo de medir o desempenho dos ACs e suas adaptações com busca local. Os parâmetros modificados para cada simulação foram: Tamanho da população (Tam Pop), Número de gerações (Num Ger) e Número de repetições (Num Rep). A quantidade máxima de avaliações que é o produto do Tam Pop por Num Ger, foi mantido no valor de 10.000 e 30.000 avaliações por cada repetição para funções dependendo do cenário a ser utilizado.

3.1 Funções de benchmark

As funções de benchmarks são bastante utilizadas quando se deseja medir o desempenho de algoritmos. Essas funções são utilizadas para comparar o Algoritmos Culturais (ACs) e duas propostas de hibridização, ACs com *Simulated Annealing* (SA) e ACs com Busca Tabu (BT).

Os três algoritmos foram testados em oito funções de benchmarks do CEC2017 [21]. As definições de espaço de busca $[-100.0, 100.0]^D$ e das dimensões D=10 e D=30, são aplicadas em quatro funções básicas conforme mostra a tabela 1 e em quatro funções hibridas conforme mostra a tabela 2:

Tabela 1. Funções Básicas de benchmark

Função		Descrição e Expressão
F1	Função Bent Cigar	$f_1(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2,$
F3	Função Zakharov	$f_3 = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5x_i)^2 + (\sum_{i=1}^{D} 0.5x_i)^4,$
F5	Função Rastrigin's	$f_5(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$
F15	Função Griewank's	$f_{15}(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$

Fonte: Adaptada de [21].

Tabela 2. Funções Hibridas de benchmark

Função	Dom. Funções Básicas	Descrição	Funções Básicas
FH1	P = [0.2, 0.4, 0.4]	Função Hibrida 1	g1: Função Zakharov g2: Função Rosenbrock g3: Função Rastringin's
FH4	P = [0.2, 0.2, 0.2, 0.4]	Função Hibrida 4	g_1: Função Elíptica condicionada Alta g_2: Função Ackley's g_3: Função Schaffer's g_4: Função Rastrigin's
FH5	P = [0.2, 0.2, 0.3, 0.3]	Função Hibrida 5	g_1: Função Bent Cigar g_2: Função HGBat g_3: Função Rastrigin's g_4: Função Rosenbrock
FH8	P = [0.2, 0.2, 0.2, 0.2, 0.2]	Função Hibrida 8	g_1: Função Elíptica condicionada Alta g_2: Função Ackley's g_3: Função Rastringin's g_4: Função HGBat g_3: Função Discus

Fonte: Adaptada de [21].

3.2 Cenário 1 e 2

Os dois cenários possuem Número de Repetições igual a 50. A cada nova repetição é gerada uma nova população aleatória com função densidade de probabilidade uniforme definida dentro dos limites de cada variável. As tabelas 3 apresenta os parâmetros dos cenários 1 e 2, relacionados com funções básica e hibridas respectivamente.

Tabela 3. Cenário 1 e 2 – Funções Básicas e Hibridas

D	Cenário1	Nome Função	Cenario2	Nome Função	Tam Pop	Num. Ger	Max. Avaliação	D*Max_FES	Num Rep
10	**F1**	**Bent Cigar**	**FH1**	**Hibrida 1**	50	200	10.000	100.000	50
10	**F3**	**Zakharov**	**FH4**	**Hibrida 4**	50	200	10.000	100.000	50
10	**F5**	**Rastrigin's**	**FH5**	**Hibrida 5**	50	200	10.000	100.000	50
10	**F15**	**Griewank's**	**FH8**	**Hibrida 8**	50	200	10.000	100.000	50

30	**F1**	**Bent Cigar**	**FH1**	**Hibrida 1**	50	600	30.000	300.000	50
30	**F3**	**Zakharov**	**FH4**	**Hibrida 4**	50	600	30.000	300.000	50
30	**F5**	**Rastrigin's**	**FH5**	**Hibrida 5**	50	600	30.000	300.000	50
30	**F15**	**Griewank's**	**FH8**	**Hibrida 8**	50	600	30.000	300.000	50

3.3 Fluxograma para o teste

A figura 4 mostras através de um fluxograma as etapas utilizadas para execução dos testes para este artigo.

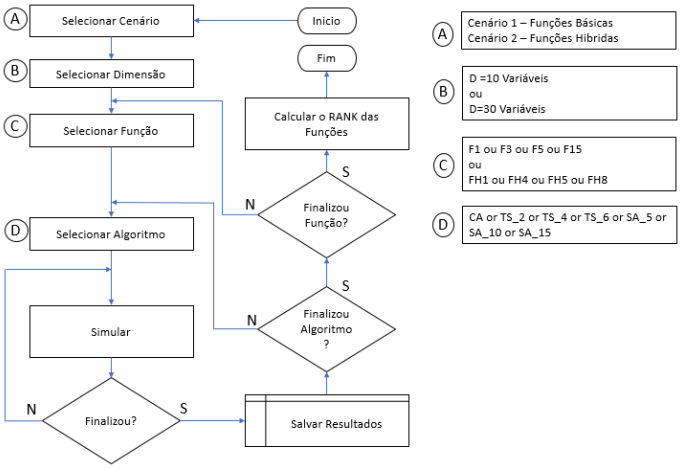

Fig. 4. Organograma para o teste.

4 Resultados

Para todos os ACs utilizados, manteve-se o mesmo padrão de seus parâmetros e foram realizados testes preliminares com a intenção de definir o tamanho da variável de energia do (SA) e do tamanho da lista tabu do (BT), e após estes testes ficou definido que três variáveis de cada técnica que se mostrassem com resultados mais diferenciados, seriam utilizadas para incrementar novos cenários de testes. Na variação dos ACs com o SA, a variável de Energia (SA) de Busca Local sofreu alterações de 5, 10 e 15. Na variação dos ACs com o BT, ocorre a variação do tamanho da lista tabu em 2, 4 e 6. Dessa maneira, surgem três novas condições de testes para este algoritmo hibrido. Ficando, então com sete variações de algoritmos. A tabela 4 apresenta cada uma das alternativas.

Table 4. Algoritmos para teste

Algoritmo	Descrição
AC	Algoritmo Cultural Clássico
SA_5	Algoritmo Cultural + *Simulated Annealing* (Energia de Busca Local = 5)
SA_10	Algoritmo Cultural + *Simulated Annealing* (Energia de Busca Local = 10)

SA_15	Algoritmo Cultural + *Simulated Annealing* (Energia de Busca Local = 15)
BT_2	Algoritmo Cultural + Busca Tabu (Tamanho da Lista Tabu = 2)
BT_4	Algoritmo Cultural + Busca Tabu (Tamanho da Lista Tabu = 4)
BT_6	Algoritmo Cultural + Busca Tabu (Tamanho da Lista Tabu = 6)

Para avaliação, os resultados dos melhores valores de cada simulação foram submetidos aos testes de Friedman, Friedman Aligned e Quade. Os testes foram realizados com os resultados de D=10, D=30 e com a junção destes dois resultados. As entradas de dados para análise estão na tabela 5.

Table 5. Entrada de dados de D-10 e D-30

Data-set	AC	BT2	BT4	BT6	SA5	SA10	SA15
10-F1	1,25E-13	1,05E-13	3,90E-14	7,60E-14	1,04E-13	2,50E-14	2,90E-14
10-F3	0	0	0	0	0	0	0
10-F5	1,66E-12	1,74E-12	4,38E-12	4,54E-12	4,46E-11	9,12E-12	1,00E-11
10-F15	0,004932	0,009857	7,87E-12	1,23E-10	0,004932	7,44E-13	0,007396
10-FH1	0,053469	0,066624	0,041932	0,000210	0,126893	0,001537	0,142696
10-FH4	1,069561	1,069561	1,069561	1,069561	1,069561	1,069561	1,069561
10-FH5	0,524180	0,300677	0,383617	0,480464	0,358352	0,588176	0,396050
10-FH8	1,569561	1,569561	1,569561	1,569561	1,569561	1,569561	1,569561
30-F1	6,27E-08	7,25E-08	4,71E-08	1,11E-07	3,65E-08	1,41E-08	3,14E-08
30-F3	0	0	0	0	0	0	0
30-F5	0,000221	1,46E-05	4,83E-05	0,000447	6,07E-05	0,000495	0,002175
30-F15	8,38E-06	4,22E-06	9,90E-06	9,62E-06	5,92E-06	7,03E-06	9,23E-06
30-FH1	0,053469	0,066624	0,041932	0,000210	0,126893	0,001537	0,142696
30-FH4	1,069561	1,069561	1,069561	1,069561	1,069561	1,069561	1,069561
30-FH5	0,524180	0,300677	0,383617	0,480464	0,358352	0,588176	0,396050
30-FH8	1,569561	1,569561	1,569561	1,569561	1,569561	1,569561	1,569561

Os dados da tabela 5 foram submetidos a avaliação dos testes tratados neste trabalho. Obteve-se como resultados os dados mostrados na tabela 6, 7 e 8. Os valores em vermelho são os melhores resultados em cada teste associado ao algoritmo que teve melhor desempenho para o conjunto de funções.

Tabela 6. Resultado para D-10

Algoritmo	Friedman	Friedman Aligned	Quade
AC	3,375	22,875	3,277778
BT2	4,3125	27,3125	4,416667
BT4	4,25	37,375	4,694444
BT6	4,1875	30,4375	4,472222
SA5	3,6875	23,0625	3,666667
SA10	4,9375	33,4375	4,666667
SA15	3,25	25	2,805556

Tabela 7. Resultado para D-30

Algoritmo	Friedman	Friedman Aligned	Quade
AC	3,375	23,75	3,25
BT2	5,1875	31,8125	5,430556
BT4	3,625	33,375	4,083333
BT6	3,3125	25,0625	3,513889
SA5	4,6875	33,8125	4,708333
SA10	4,5625	29,3125	4,125
SA15	3,25	22,375	2,888889

Sendo, a tabela 6, o resultado obtido quando se aplicou somente o conjunto de dados com dimensão 10. Nesta tivemos o SA15 como algoritmo melhor classificado nos testes Friedman e Quade. Já no teste Friedman Aligned, o AC obteve melhor classificação.

A tabela 7 mostra os valores resultantes dos testes para o conjunto de dados com dimensão 30 em cada problema. Sendo neste cenário o SA15 foi o melhor em todos os testes.

Tabela 8. Resultado para a junção de D-10 e D-30

Algoritmo	Friedman	Friedman Aligned	Quade
AC	3,375	45,4375	3,220588
BT2	4,75	58,0625	4,886029
BT4	3,9375	70,5	4,389706
BT6	3,75	54,875	3,974265
SA5	4,1875	57,25	4,238971
SA10	4,75	62,375	4,430147
SA15	3,25	47	2,860294

Após avaliação individual dos algoritmos propostos com os devidos problemas, resolveu-se juntar as bases de dados geradas com dimensões 10 e 30, para verificar se a classificação destes três testes se diferencia muito dos resultados anteriores. A tabela 8 mostra o resultado após a submissão destes dados aos testes aqui abordados. Para o resultado da união dos valores obtidos notou-se que este acompanhou a simulação para dimensão 10.

5 Conclusão

O uso dos testes de Friedman, Friedman Aligned e Quade, ajudaram na avaliação das variações dos algoritmos propostos, evitando a comparação de pares de algoritmos. O conjunto de dados de D=10 e de D=10+30 (união dos resultados de testes de comportamento de algoritmos com domínio 10 com os resultados de testes com domínio igual a 30), obtiveram resultados iguais quanto aos melhores algoritmos (SA15 e AC) para as funções utilizadas. Porém, com os dados de D=30, o resultado foi único apontando o SA15 como melhor classificado nas simulações. Em relação ao AC com Busca Tabu, o melhor resultado foi encontrado com a lista tabu de tamanho 6.

Foi acrescido ao espaço de crenças do AC, especificamente no conhecimento topográfico, os valores posicionais dos 3 melhores valores encontrados nas buscas locais criando desta forma uma região de resultados promissores para avaliação do AC. Além disso, nota-se que a hibridização do AC com as técnicas de busca local tende a obter melhores resultados na solução de funções multivariáveis. Os testes usados para esta avaliação são de fácil implementação e robusto nos resultados quando pouco se sabe sobre o problema.

Referências

1. Bray, D. and S. Lay, *Computer simulated evolution of a network of cell-signaling molecules.* Biophysical journal, 1994. **66**(4): p. 972-977.
2. Adami, C., *The use of information theory in evolutionary biology.* Annals of the New York Academy of Sciences, 2012. **1256**(1): p. 49-65.

3. De Jong, K., D. Fogel, and H.-P. Schwefel, *Handbook of Evolutionary Computation.* IOP Publishing Ltd and Oxford University Press, 1997.
4. Eiben, A.E. and J.E. Smith, *Introduction to evolutionary computing*. Vol. 53. 2003: Springer.
5. Petrowski, A. and S. Ben-Hamida, *Evolutionary Algorithms*. 2017: John Wiley & Sons.
6. Gaspar-Cunha, A., R. Takahashi, and C.H. Antunes, *Manual de computação evolutiva e metaheurística*. 2012: Imprensa da Universidade de Coimbra/Coimbra University Press.
7. Reynolds, R.G. *An introduction to cultural algorithms*. in *Proceedings of the third annual conference on evolutionary programming*. 1994. Singapore.
8. Wei, Z., B. Yan-ping, and Z. Ye-qing. *The application of an improved cultural algorithm in grid computing*. in *Control and Decision Conference (CCDC), 2013 25th Chinese*. 2013. IEEE.
9. Blum, C., et al., *Hybrid metaheuristics in combinatorial optimization: A survey.* Applied Soft Computing, 2011. **11**(6): p. 4135-4151.
10. Derrac, J., 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, 2011. **1**(1): p. 3-18.
11. Friedman, M., *A comparison of alternative tests of significance for the problem of m rankings.* The Annals of Mathematical Statistics, 1940. **11**(1): p. 86-92.
12. Derrac, J., et al., *Analyzing convergence performance of evolutionary algorithms: A statistical approach.* Information Sciences, 2014. **289**: p. 41-58.
13. Demšar, J., *Statistical comparisons of classifiers over multiple data sets.* Journal of Machine learning research, 2006. **7**(Jan): p. 1-30.
14. Krohling, R.A., R. Lourenzutti, and M. Campos, *Ranking and comparing evolutionary algorithms with Hellinger-TOPSIS.* Applied Soft Computing, 2015. **37**: p. 217-226.
15. Moscato, P. and C. Cotta, *A modern introduction to memetic algorithms*, in *Handbook of metaheuristics*. 2010, Springer. p. 141-183.
16. Reynolds, R.G. and Y.A. Gawasmeh. *Evolving heterogeneous social fabrics for the solution of real valued optimization problems using cultural algorithms*. in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. 2012. IEEE.
17. Ali, M.Z., N. Awad, and R.G. Reynolds. *Balancing search direction in cultural algorithm for enhanced global numerical optimization*. in *Swarm Intelligence (SIS), 2014 IEEE Symposium on*. 2014. IEEE.
18. Silva, D.J.A., *Algoritmos Culturais com Abordagem Memética e Multipopulacional Aplicados a Problemas de Otimização*, in *Instituto de Tecnologia*. 2012, Universidade Federal do Pará: Belém-PA. p. 134 p.
19. Gendreau, M. and J.-Y. Potvin, *Tabu search*, in *Search methodologies*. 2014, Springer. p. 243-263.
20. García, S., et al., *A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization.* Journal of Heuristics, 2009. **15**(6): p. 617-644.
21. Awad, N., et al., *Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization.* 2016.