Ciências da Informação 291 (2015) 115-127

Listas de conteúdos disponíveis em ScienceDirect

Ciências da Informação

página inicial do jornal: www.el sevier .com / locate / ins

Sobre o esclarecimento de equívocos ao comparar variantes do Arti fi cial Bee Colony Algorithm, oferecendo uma nova implementação

Marjan Mernik uma , ⇑ , Shih-Hsi Liu b , Dervis Karaboga c , Matej Črepinšek uma

uma Universidade de Maribor, Faculdade de Engenharia Elétrica e Ciência da Computação, Smetanova 17, 2000 Maribor, Eslovênia
b California State University, Fresno, Departamento de Ciência da Computação, 2576 E San Ramon Dr, Fresno, CA 93740, EUA
c Universidade Erciyes, Departamento de Engenharia da Computação, 38039 Melikgazi, Kayseri, Turquia

informações do artigo

Historia do artigo:
Recebido em 20 de dezembro de 2013
Recebido na forma revisada em 21 de julho de 2014
Aceito em 15 de agosto de 2014
Disponível online em 28 de agosto de 2014

Palavras-chave:
Inteligência de enxame
Replicação de experimento de algoritmo de colônia de abelhas artificial

abstrato

Arti fi cial Bee Colony (ABC) é um algoritmo de Swarm Intelligence que obteve a atenção e o favor de pesquisadores da metaheurística nos últimos anos. Compreende um bom equilíbrio entre a exploração (fase de abelha empregada e fase de abelha observadora) e exploração (fase de abelha escuteira). Como hoje em dia, mais pesquisadores estão usando o ABC e suas variantes como grupo de controle para realizar comparações, é crucial que as comparações com outros algoritmos sejam justas. Este artigo aponta para alguns equívocos ao comparar algoritmos meta-heurísticos baseados em iterações (gerações ou ciclos) com ênfase especial no ABC. Esperamos que, por meio de nossas descobertas, este artigo possa ser tratado como um farol para lembrar os pesquisadores de aprender com esses erros.

1. Introdução

Uma recomendação comum para comparação justa entre algoritmos meta-heurísticos [12] (por exemplo, Algoritmos Evolucionários [6,26]) para funções de referência construídas artificialmente é comparar algoritmos com base em um número igual de avaliações de aptidão consumidas [9,25] . No entanto, em experimentos com algoritmos meta-heurísticos, os pesquisadores geralmente empregam uma condição de parada baseada no número máximo de iterações (gerações ou ciclos), e não no número máximo de avaliações de aptidão. A fim de superar esse problema, usando o mesmo número máximo de avaliações de aptidão para todos os algoritmos, embora ainda usando a implementação original de algoritmos com base em iterações, os pesquisadores geralmente usam a seguinte fórmula:
MCN ¼ FEs = NP, Onde MCN é o número máximo de iterações (gerações ou ciclos), FEs é o número máximo de avaliações de aptidão, e NP é o tamanho de uma população. No entanto, a fórmula proposta seria válida apenas se não houvesse pesquisas extras no algoritmo meta-heurístico. É bastante comum hoje em dia que algoritmos meta-heurísticos são combinados com pesquisas locais (por exemplo, Algoritmo Memético [71]) ou incorporar outros algoritmos (por exemplo, conjuntos). Portanto, a aplicação da fórmula mencionada deve ser feita após uma consideração cuidadosa dessa questão.

Arti fi cial Bee Colony (ABC) [3,48-50] é um algoritmo interessante de Swarm Intelligence (SI) com boas habilidades de exploração e aproveitamento [20,24] . Ele tem sido aplicado a vários problemas práticos, bem como a funções de otimização criadas artificialmente. Para obter uma lista detalhada do uso do ABC, consulte as pesquisas abrangentes recentes sobre o ABC [8,13,52,106] , onde pode ser observado que ABC é mais popular do que outros algoritmos de abelhas (por exemplo, Otimização de colônias de abelhas, Algoritmo de abelhas) [51] .

⇑ Autor correspondente.
Endereço de e-mail: marjan.mernik@uni-mb.si (M. Mernik), shliu@CSUFresno.edu (S.-H. Liu), karaboga@erciyes.edu.tr (D. Karaboga), matej.crepinsek @ uni-mb.si (M. Črepinšek).

http://dx.doi.org/10.1016/j.ins.2014.08.040
0020-0255 / 2014 Elsevier Inc. Todos os direitos reservados.

Curiosamente, a fase de abelhas escoteiras na implementação básica do ABC [3,48-50] usa uma avaliação de aptidão extra por iteração quando uma certa condição se mantém (ou seja, quando uma solução não pode ser melhorada após um número predeterminado de tentativas controladas pelo parâmetro 'limite'). Portanto, a fórmula MCN ¼ FEs = NP não pode mais ser usado ao aplicar o ABC básico. Este fato tem sido esquecido por muitos pesquisadores, apesar de que no código ABC publicamente disponível pode ser claramente visto que a aptidão é consumida durante a fase de escoteira devido à função de chamada iniciar ðÞ usado durante a fase de inicialização (também é óbvio que cada nova solução aleatória precisa ser avaliada):

```
void init(int index)
begin
. . .
    fitness[index] = CalculateFitness(f[index]);
    trial[index] = 0;
end
void SendScoutBees()
begin
. . .
    if (trial[maxtrialindex] >= limit) then
        init(maxtrialindex);
    end
end
```

Por outro lado, deve-se notar que durante a fase de inicialização do algoritmo ABC básico, o número de avaliações de aptidão é a metade de NP. Papel [3] deu uma aproximação do número de avaliações de aptidão consumidas do ABC básico. Os inventores do ABC escreveram em [3] : '' Uma vez que o número de espectadores é igual a SN, em cada ciclo 2/As buscas de SN são conduzidas por abelhas empregadas e abelhas observadoras. Portanto, quando o número máximo de ciclo (MCN) é atingido, totalmente, 2/SN/Pesquisas MCN são realizadas. Portanto, a complexidade de pesquisa do ABC é proporcional a 2/SN/MCN. '' Observe que no ABC o tamanho da colônia NP é 2/SN (SN abelhas empregadas + SN espectadores). Claramente, esta fórmula é apenas uma aproximação e não pode ser usada diretamente em comparação com outros algoritmos meta-heurísticos. Parece que muitos pesquisadores simplesmente negligenciaram a palavra 'proporcional' na declaração acima mencionada e literalmente tomaram apenas a fórmula FEs = 2 SN MCN. Na implementação básica do ABC, em comparação com alguns outros algoritmos (por exemplo, DE [91]), a determinação estática exata do número de avaliações de aptidão não pode ser definida com base em NP e MCN com antecedência (o scout bee pode ou não ser empregado em uma iteração particular, que depende do valor do parâmetro de controle 'limite'). Neste artigo, o limite superior das avaliações de aptidão no ABC básico é definido e dado como: FEs <= (SN + SN + 1)/MCN + SN. Em cada iteração existem SN abelhas empregadas, SN abelhas observadoras e no máximo 1 abelha batedora. Durante a fase de inicialização, adicionais SN soluções são criadas e avaliadas. Claro, usar uma aproximação (até mesmo um limite superior) para o número de avaliações de aptidão enquanto compara diferentes algoritmos meta-heurísticos muitas vezes não é uma opção boa e satisfatória, pois sempre haverá algumas ameaças à validade de nossas conclusões. No entanto, o problema pode ser facilmente resolvido mudando ligeiramente a implementação ABC (ver Algoritmos 1 e 2 dentro Apêndice A):

definir a condição de parada com o número máximo de avaliações de aptidão e contando as avaliações de aptidão dinamicamente (durante a execução ABC) na função calcularFitness ðÞ usado em todas as fases ABC (inicialização, fase de abelha empregada, fase de abelha observadora e fase de abelha escuteira) e
alcançar uma abelha empregada que se tornará uma abelha escuteira não requer uma avaliação de aptidão na fase de abelha empregada, pois a solução será substituída por uma solução aleatória de qualquer maneira. Em tal implementação, haverá exatamente 2/SN avaliações de aptidão por iteração (geração ou ciclo).

Como mais e mais pesquisadores estão usando variantes ABC hoje em dia, é crucial que a comparação com outros algoritmos seja justa. Omitir a contagem de avaliações de aptidão durante a fase de abelha escoteira é um problema onipresente de muitas variantes do ABC. Devido a esse fato, muitas conclusões que resultaram desses experimentos podem ter sérias ameaças à validade, e algumas delas podem agora ser inválidas, particularmente quando as avaliações de aptidão extra são consumidas nos buscadores locais combinados com o algoritmo ABC. Observe que nada pode ser generalizado e as comparações precisam ser revisadas caso a caso. Os principais objetivos deste artigo são:

1. Oferecer implementações alternativas de ABC para fixar as avaliações de aptidão total para um número constante: (a) definição da condição de parada com base na contagem de cada avaliação de aptidão durante todas as fases; (b) reimplementação do ABC, onde o número de avaliação de aptidão total é definido em termos de MCN e SN, mas sob o
   condição de que o número de avaliações de aptidão consumidas não seja maior do que o permitido.
2. Estudar mais de 100 artigos sobre ABC (variantes), enquanto verifica:
   (a) se o algoritmo / pseudo-código / fluxograma / descrição ABC (variante) indica claramente que as avaliações de aptidão foram
   devidamente contado durante a fase de abelha escuteira;
   (b) se a comparação de ABC (variante) com outros algoritmos foi baseada em igual número de avaliações de aptidão consumidas;

(c) se os gráficos de convergência comparando ABC (variante) com outros algoritmos foram baseados em igual número de avaliações de aptidão ações consumidas; e
3. Investigue se as conclusões dos experimentos em [49] ainda se manterá se os experimentos forem replicados levando em consideração o mesmo número de avaliações de aptidão para ambas as implementações oferecidas.

O artigo está organizado da seguinte forma. Seção 2 mostra a investigação do ABC e suas variantes junto com nossos achados, seguido por replicações de alguns experimentos do ABC na seção 3 . As observações finais são apresentadas na Seção 4 . Os pseudocódigos ABC de duas novas implementações são apresentados em Apêndice A . Ambos os algoritmos garantem que não mais do que o número máximo permitido de avaliações de aptidão sejam consumidas.

2. Investigação do uso de ABC em alguns papéis existentes

Para ter uma boa visão geral do uso existente do ABC, realizamos uma bola de neve para a frente [56] que é um processo durante os estudos de mapeamento sistemáticos [78] . Como bola de neve para a frente, os jornais que citaram o jornal original da ABC [49] foram pesquisados, estudados e classificados. No geral, papel [49] tem sido por um longo período de tempo o artigo mais citado na revista Applied Soft Computing. A pesquisa Scopus em 22 de agosto de 2013 retornou 567 ocorrências de artigos publicados no período de janeiro de 2009 a agosto de 2013. No entanto, apenas 108 artigos estavam disponíveis eletronicamente para nós. Entre esses, 89 artigos [1,2,4,5,7,11,14-17,19,22,27-32,34-47,53-55,58-60,62-70,73-77,79,80,82 –90,92–102,104,105,107–119] foram relevantes para o nosso estudo (ABC ou sua variante foram explicados nos artigos ou usados para comparações), enquanto o artigo do periódico ABC original [49] só foi mencionado nos 19 artigos restantes em trabalhos relacionados ou trabalhos futuros. Estamos cientes do fato de que não estudamos todos os artigos do ABC e que as últimas conclusões podem ter algumas ameaças de validade devido a esse fato. Para fortalecer a validade de nossas conclusões, um intervalo de confiança (margem de erro) [18] foi adicionado com um nível de confiança de 95%. O intervalo de confiança em nosso estudo foi de 8,5. As seguintes questões de pesquisa foram feitas:

1. Quantos artigos claramente indicaram no pseudocódigo / fluxograma / descrição ABC que as avaliações de aptidão foram consumidas na fase de abelha escoteira?
2. Quantos artigos usaram a condição de igual número de avaliações de aptidão consumidas para comparação ABC?
3. Como muitos artigos apresentaram gráficos de convergência comparando ABC com outros algoritmos com base em igual número de avaliações de aptidão consumidas?

Depois de ler e estudar esses 89 artigos, os resultados foram bastante surpreendentes e exemplificaram o quão onipresente o problema discutido na Seção 1 é:

Embora em código ABC publicamente disponível (http://mf.erciyes.edu.tr/abc) houve uma indicação clara de que as avaliações de aptidão física foram consumidas durante a fase de abelha escoteiro, apenas alguns autores (por exemplo, [68,82,95,108,113]) também indicaram isso em seu trabalho. Houve 11 estudos (12,4%), enquanto em 18 estudos (20,2%) a resposta foi indecisa, pois não havia detalhes suficientes nesses estudos. Nos 60 estudos restantes (67,4%), não houve indicação de que as avaliações de aptidão física tivessem sido contadas na fase de escoteira. Claro, há uma chance de que apenas a descrição ABC não seja precisa o suficiente, mas sua implementação corrigiu o problema. Usando o intervalo de confiança, pode-se concluir, usando o nível de confiança de 95%, que 67: 4 8: 5% dos trabalhos examinados não tiveram avaliações de aptidão contadas corretamente.

Comparar algoritmos meta-heurísticos com base em um número igual de iterações só é adequado se todos os algoritmos comparados consumiram um número igual de avaliações de aptidão em cada iteração (ver discussão na Seção 1) Na verdade, 36 estudos de 89 (40,4%) usaram comparações inadequadas com base em números iguais de iterações, enquanto 28 estudos (31,5%) compararam algoritmos com base em um número igual de avaliações de aptidão ou no tempo de CPU. Nos 25 estudos restantes (28,1%), não havia detalhes suficientes para descobrir como as comparações foram feitas. Neste último caso, parece que em muitos estudos apenas a melhor solução era de interesse, independentemente dos custos computacionais. Para resolver problemas práticos difíceis, onde os resultados em tempo real não são necessários, esse pode ser um caminho viável. Observe que, apesar do fato de que mais de 30% dos estudos tentaram comparar de forma justa os algoritmos com base em igual número de avaliações de aptidão, isso não significa que as comparações foram de fato justas, pois pode haver alguns casos em que as avaliações de aptidão não foram contadas (por exemplo, na fase de escuteira ou na fase adicional de busca local). Usando o intervalo de confiança, pode-se concluir, usando o nível de confiança de 95%, que 40: 4 8: 5% dos trabalhos examinados estavam usando comparações inadequadas com base em números iguais de iterações.

Deve estar claro agora que comparar em um número igual de iterações é principalmente inapropriado para algoritmos meta-heurísticos (apenas em casos especiais). O mesmo é verdade para gráficos de convergência que não devem ser baseados em um número igual de iterações se algoritmos comparados não consumirem um número igual de avaliações de aptidão em cada iteração. Ao ler 89 estudos sobre o ABC, frequentemente encontramos afirmações sobre novas versões do ABC com a propriedade de convergência extremamente rápida. Mas, a convergência foi medida corretamente com base em avaliações de aptidão ou no tempo de CPU? De 89 estudos, apenas 44 incluíram discussões sobre gráficos de convergência, e 28 estudos de 44 (63,6%) ainda estavam usando gráficos de convergência comparando ABC e outros algoritmos com base em um número igual de iterações, enquanto apenas 16 estudos de 44 (36. 4%) estavam usando gráficos de convergência com base no número de avaliações de aptidão ou no tempo de CPU. Usando o intervalo de confiança, pode-se afirmar usando o nível de confiança de 95% que 63: 6 8: 5% do trabalho examinado não interpretou corretamente os gráficos de convergência.

Após a análise quantitativa, é informativo examinar alguns detalhes interessantes de estudos específicos. Freqüentemente, os pesquisadores estão convencidos de que o projeto e a execução do experimento estão corretos. A partir deste estudo, não tivemos a impressão de que um design errado foi usado deliberadamente. Os pesquisadores simplesmente não estão cientes desse problema. É bastante comum que os pesquisadores simplesmente acreditem que a comparação em um número igual de iterações é justa e expressem essa crença em seus artigos. Alguns exemplos usando ABC são:

'' Para fazer uma comparação justa, o número máximo de geração e a faixa de pesquisa dos parâmetros são os mesmos para todos os algoritmos. Ou seja, o número máximo de geração é definido como 100 e o tamanho da população é definido como 120 [102] . '' Mas, isso está longe de ser justo neste caso particular [102] . Em cada iteração, os algoritmos comparados em [102] (por exemplo, HTCMIABC, CLONALG e ABC) consumiu um número diferente de avaliações de aptidão por indivíduo (ver Fig. 2 em [102]) Com base em tal experimento, os autores [102] concluiu: '' É óbvio a partir da Fig. 3 [102] que a velocidade de convergência do HTCMIABC é muito rápida e todos os parâmetros a, bec convergem para os valores verdadeiros correspondentes rapidamente (menos de 30 gerações). Portanto, o HTCMIABC é muito eficiente para a identificação de parâmetros de um sistema caótico [102] . '' A conclusão pode ser válida, mas um experimento mal planejado não pode apoiar ou provar tal conclusão.

'' Para os parâmetros de ABC, deixe o número da iteração ser 500 para todas as funções abaixo para ABC enquanto as iterações de QPSOs individuais são 1000. Deixe ABC e QPSO rodarem 100 vezes independentemente com 50 abelhas empregadas e 50 partículas, respectivamente [28] . '' Neste caso particular, o número de avaliações de aptidão para QPSO é: FEs (QPSO) = num_ partículas/num_iterations = 50/1000 = 50.000, enquanto usa a fórmula de [3] dando também FEs (ABC) = (2/empregadas_bees)/num_iterations = (2/50)/500 = 50.000. Infelizmente, a fórmula de [3] são apenas uma aproximação. Apesar de uma intenção fiel, uma comparação justa não foi alcançada em [28] .

'' A contagem de iterações não é mais uma medida razoável, pois diferentes complexidades computacionais podem ser tomadas em cada iteração para diferentes algoritmos. Para comparar os diferentes algoritmos, um método de medida justo deve ser selecionado. Neste artigo, usamos o número de avaliações de função (FEs) como um critério de medida. Todos os algoritmos foram encerrados após 100.000 FEs [110] . '' O experimento estaria correto se todas as avaliações de aptidão física tivessem sido contadas corretamente. Infelizmente, não há nenhuma indicação de que as avaliações de aptidão tenham sido contadas durante a fase de abelhas escoteiras e no operador de crossover recém-sugerido de Hybrid Arti fi cial Bee Colony (HABC) em [110] .

O ABC original usava no máximo uma abelha batedora por iteração [49] . O problema da contagem inadequada de avaliações de aptidão em a fase de abelhas escoteiras é extravagante no sentido de que mais abelhas escoteiras são usadas por iteração. Na verdade, houve muitas sugestões para a melhoria do ABC que inclui mais exploração do espaço de busca usando abelhas batedoras extras. Alguns dos estudos incluídos que usaram mais de uma abelha batedora por iteração são: [7,27,41,67,69,80,88,90,93,96,97,104,105, 107.113.117] . Nesses trabalhos, as avaliações de aptidão para a fase de escoteira podem ou não ser contadas para as avaliações cumulativas de aptidão. Além disso, em alguns casos, o número de abelhas observadoras não é o mesmo que o número de abelhas empregadas (por exemplo, [69]) Portanto, o número correto de avaliações de aptidão, neste caso, deve ser examinado cuidadosamente.

Além da fase de escoteiros do ABC, um problema semelhante também pode surgir quando as pesquisas locais são incluídas no ABC, gastando ainda mais avaliações de aptidão por iteração. Alguns dos estudos incluídos que sugeriram pesquisas locais adicionais no ABC são: [42,45-47,69,77,83,96,100,109,117] . Observe que, nesses estudos, as avaliações de aptidão extra consumidas na busca local nem sempre foram contadas. Isso pode ou não invalidar suas conclusões. Por exemplo, em [77,83,100] a comparação entre ABC e outros algoritmos é feita em tempo de CPU. Nesse caso, a contagem adequada das avaliações de aptidão é de menor importância. Suas conclusões de tais experimentos planejados ainda são válidas. No entanto, esses estudos podem não ser usados em comparações futuras, pois seus resultados dependem muito do hardware específico usado. Nesses casos, é mais difícil obter o mesmo ambiente experimental para uma comparação dos resultados. O outro problema pode ser que o tempo relatado às vezes também inclui o tempo gasto em algumas outras tarefas (por exemplo, coleta de lixo, tarefa do sistema operacional). Alguns exemplos de inclusão de etapas extras no ABC são descritos vividamente em:

[38] onde EABC foi proposto (ver Fig. 1 em [38]): '' Em suma, para obter o EABC melhorado, reforçamos cada fase do algoritmo ABC adicionando as características do algoritmo PSO: cada abelha recebeu uma velocidade, como se fosse uma abelha real, para permitir a pesquisa em sua própria direção e em seu próprio ritmo. Processos de atualização de velocidade foram fornecidos para abelhas iniciais, abelhas empregadas e em abelhas observadoras; assim, o EABC pode ser considerado como tendo aprimorado o processo de solução. '' Em outras palavras, após a inicialização, durante a fase de abelha empregada e a fase de abelha observadora, as avaliações de aptidão extra foram consumidas por abelhas em movimento de acordo com as regras de Otimização de Enxame de Partículas (PSO). Portanto, em cada ciclo foram consumidas duas vezes mais avaliações de aptidão (além das avaliações de aptidão extra durante a inicialização). Em tais casos, a comparação deve ser feita em igual número de avaliações de aptidão (ou no tempo de CPU como foi feito em [77,83,100]) Mas, os autores usaram o mesmo número de ciclos: '' Para uma comparação justa, todos os parâmetros indeterminados são baseados na execução do ciclo de 2000. ''

[107] onde ERABC foi proposta pela mudança da fase do observador: '' Produzir duas novas fontes alimentares candidatas V_{eu} e V'_{eu} para a abelha empregada correspondendo à fonte de alimento X_{eu} usando Eqs. (...) e (...), respectivamente. Selecione a melhor solução de $f V_{eu}; V'_{eu}; X_{eu}$ g. '' Claramente, uma avaliação de aptidão adicional foi consumida na fase de abelha observadora em relação ao ABC original. A ERABC propôs também uma busca caótica especial para a fase de abelha exploradora, gastando ainda mais avaliações de aptidão por ciclo. Apesar dos autores [107] acreditam que o design do experimento é justo, este não é o caso: '' As configurações dos parâmetros são

iguais aos de [. . .] para apenas comparar o desempenho do ABC, PSABC [. . .] junto com ERABC em todas as funções de teste. Em outras palavras, o número máximo de ciclos MCN é definido como 1000, e o número de tamanho da população SN é considerado como 40 e o limite do parâmetro de controle é igual a 100 para todas as funções de teste. "

Além disso, além da fase de abelha escoteira do ABC e várias pesquisas locais, há outra possível advertência para a contagem incorreta de avaliações de aptidão na fase de inicialização. De fato, [30-32] integrou uma busca caótica e uma abordagem baseada em oposição na fase de inicialização, consumindo duas vezes mais avaliações de aptidão do que outras abordagens para a população inicial.

Usar mais avaliações de aptidão por iteração do que outros algoritmos competitivos também invalida a comparação do gráfico de convergência entre os algoritmos comparados. É claro que a convergência para uma solução usando mais avaliações de aptidão por iteração será mais rápida. Conseqüentemente, os gráficos de convergência comparando diferentes algoritmos meta-heurísticos devem ser baseados no número de avaliações de aptidão ou tempo de CPU (o último é menos desejável devido a dificuldades em comparações futuras). Um exemplo onde os pesquisadores relataram convergência rápida da variante ABC é dado em [58] (ver Fig. 1 em [58]): " No algoritmo PS-ABC, cada abelha empregada ou abelha observada poderia prever suas próximas soluções por três diferentes equações de busca de solução e selecionar a melhor. Então a abelha pode decidir onde deve ir para explorar ou explorar. Os resultados da simulação mostram que o PS-ABC pode não apenas ter as vantagens do I-ABC, mas também possuir os lados positivos do ABC e do GABC. Ou seja, o PS-ABC possui uma velocidade de convergência extremamente rápida como o I-ABC e um desempenho de pesquisa muito bom. " Nesse caso, comparar um número igual de iterações é obviamente inapropriado. Gráficos de convergência (Figs. 2–15 em [58]) comparando ABC, GABC, I-ABC e PS-ABC não mostram velocidade de convergência real. Conclusão derivada em [58] : " Os resultados indicam que a capacidade de pesquisa do PS-ABC foi melhorada. Além disso, o novo algoritmo híbrido mostra a convergência muito rápida como I-ABC " baseiam-se em gráficos de convergência errados e, portanto, são pelo menos questionáveis.

Finalmente, em vários artigos (por exemplo, [7,11,31,32,89,93,97,104,105]) algumas pequenas inconsistências com a implementação ABC original foram encontradas em relação à fase de abelha escoteira. Enquanto em [49] a condição para a abelha escuteira foi declarada como ð max ð tentativas eu Þ> ¼ limite º nos artigos acima mencionados a condição não era a mesma (ð max ð tentativas eu Þ> limite º ou ð max ð tentativas eu Þ ¼ limite º). Esta é uma variação do algoritmo ABC original e os leitores só precisam estar cientes desse fato.

Em resumo, o estudo de alguns trabalhos existentes usando ABC revela que a comparação entre variantes ABC e outros algoritmos é menos do que satisfatória. Isso está de acordo com um estudo recente [61] onde foi mostrado que para problemas de alta dimensão, com parâmetros de controle sintonizados e busca local, o ABC original funciona tão bem quanto as variantes ABC.

3. Replicação de alguns experimentos ABC

A fim de mostrar como a comparação com base em um número igual de iterações, mas com um número diferente de avaliações de aptidão consumidas, pode ou não influenciar as conclusões tiradas, replicamos o experimento do artigo ABC original [49] . A fim de avaliar o desempenho do ABC seus inventores em [49] fez um pequeno experimento nas funções clássicas de benchmark apresentadas em [57] , onde os resultados para DE, PSO e EA foram mostrados nas seguintes funções de benchmark numéricas: função Schaffer bidimensional f Função de esfera 1, 5-dimensional f 2, função Griewank 50-dimensional f Função de Rastrigina 3, 50-dimensional f Função de Rosenbrock 4 e 50-dimensional f 5. Em comparação [57] DE, PSO e EA aplicados
100.000 avaliações de aptidão para funções f 1— f 2, e 500.000 avaliações de aptidão para f 3— f 5. Para replicar o experimento de [57] e os resultados comparados com ABC, a seguinte configuração de parâmetro de controle foi usada em [49] : tamanho da colônia ¼ 100 (abelhas empregadas ¼ 50; abelhas espectadoras ¼ 50), MCN ¼ 1000 para f 1— f 2 e MCN ¼ 5000 para f 3— f 5, e corre ¼ 30. No entanto, essa configuração é apenas uma aproximação, pois em cada iteração (ciclo) uma avaliação de aptidão extra pode ser aplicada para abelha escoteira, além das avaliações de aptidão consumidas durante a fase de inicialização. Observe que apenas aqueles parâmetros de controle foram mencionados, o que influenciou o número de avaliações de aptidão (outros podem ser encontrados em [49] e foram aplicados também em nosso trabalho). O objetivo de nossa replicação do experimento em [49] foi verificar se as conclusões ainda eram válidas se todos os algoritmos comparados (DE, PSO, EA e ABC) usaram números iguais de avaliações de aptidão levando em consideração que avaliações extras de aptidão podem ser consumidas durante a fase de escoteira, além daquelas também consumido durante a fase de inicialização. Os resultados obtidos pela primeira implementação, a condição de parada depende da avaliação de aptidão (Algoritmo 1 a partir de Apêndice A), e para a segunda implementação (Algoritmo 2 a partir de Apêndice A), cuja condição de parada é definida em termos de MCN e SN, são apresentados em tabela 1 (observe que, por razões práticas, valores menores que E 12

foram relatados como 0 em [49] bem como em tabela 1) Comparando os resultados obtidos em [49] (coluna ABC [49] dentro tabela 1) e nossa replicação do experimento com
ABC (coluna ABC (MCN) dentro tabela 1) podemos concluir que os resultados para f 1— f 4 são iguais, embora haja uma pequena, mas insignificante diferença na f 5. No entanto, também pode ser observado que o número de avaliações de aptidão (tabela 1) para f 1— f 2 e para
f 3— f 5 não eram 100.000 e 500.000, respectivamente, conforme assumido em [49] e realmente usado em [57] . ABC consumiu até 1.046 avaliações de aptidão extra para f 1 e tão baixo quanto 50 avaliações extras para f 5. O número de avaliações de aptidão extra depende fortemente da configuração do parâmetro de controle ABC 'limite', que foi definido em [49] como limit = Employee_bees/D. Portanto, a solução foi abandonada em cada ciclo após 50/2 = 100 tentativas em f 1, 50/5 = 250 tentativas em f 2 e 50/50 = 2500 tentativas em
f 3— f 5. Isso explica o número muito diferente de avaliações de aptidão extra na fase de abelha escoteira para f 1— f 5. No entanto, nosso objetivo neste ponto não era encontrar a melhor configuração para o parâmetro de controle 'limite', mas replicar exatamente o experimento original em [57] . Portanto, executamos ABC usando duas implementações de Apêndice A (Algoritmos 1 e 2), que obedecia estritamente a um

tabela 1
Replicação de experimentos.

f	abc [49]	ABC(MCN)	#FEs	abc criança levada 1 (FEs)	#FEs	abc criança levada 2 (FEs)	#FEs
f 1	0 ± 0	0 ± 0	101.064,2 ± 0,49	0 ± 0	100.000	0 ± 0	99.950
f 2	0 ± 0	0 ± 0	100.263,7 ± 3,62	0 ± 0	100.000	0 ± 0	99.950
f 3	0 ± 0	0 ± 0	500.100,0 ± 0,0	0 ± 0	500.000	0 ± 0	499.950
f 4	0 ± 0	0 ± 0	500.069,26 ± 2,90	0 ± 0	500.000	0 ± 0	499.950
f 5	0,133109 ± 0,262242	0,116999 ± 0,156896	500.050,0 ± 0,0	0,168245 ± 0,225376	500.000	0,156299 ± 0,211091	499.950

condição de não consumir mais do que o máximo permitido nas avaliações de aptidão. Comparando os resultados em tabela 1 (colunas abc [49] , ABC ð MCN º; abc criança levada 1 ð FEs º, e ABC criança levada 2 ð FEs º) podemos novamente notar que não há diferença nos resultados para f 1— f 4, e uma diferença insignificante para f 5. Para testes significativos [23,33] o estatístico z- Teste [10] foi aplicado desde o resultados para média e desvio padrão foram apresentados em [49] . A hipótese nula H 0 (os valores médios são iguais: H 0: eu 1 ¼ eu 2) é aceito se o p- o valor é maior do que o nível de signi fi cância a (a ¼ 0:05 foi usado em nosso caso). Já que todos p- os valores eram maiores (os menores p- valor 0,311 estava entre ABC ð MCN º vs. ABC criança levada 1 ð FEs º) do que o nível de signi fi cância, o nulo hipótese H 0 pode ser aceito com 95% de confiança. Em outras palavras, resultados para f 5 pol tabela 1 são insignificantemente diferentes e a conclusão de [49] : " Como pode ser visto nos resultados apresentados em tabela 1 , o algoritmo ABC produz o melhor desempenho entre os algoritmos considerados na presente investigação " ainda pode ser suportado se aplicado a estas pequenas funções de benchmark em particular f 1— f 5. Este é um exemplo que, apesar da replicação inexata de experimentos entre [49,57] as conclusões tiradas ainda podem ser válidas, mas precisam ser apoiadas por experimentos adicionais, onde foi mostrado que melhores resultados foram encontrados pelo ABC não devido a avaliações de aptidão extra, mas sim devido a uma melhor exploração e exploração do espaço de busca.

Na verdade, devido a uma implementação particular (Algoritmo 2) o ABC criança levada 2 ð FEs º até consumiu um pouco menos avaliações de aptidão do que o permitido (tabela 1) como durante a fase de inicialização apenas SN ¼ NP = 2 soluções são geradas inicialmente. Para este ABC implemente mentação o experimento foi replicado mesmo sob condições mais rigorosas (menos avaliações de aptidão do que o permitido) [21] . O próximo objetivo do papel muito influente [49] foi analisar o comportamento do ABC usando diferentes tamanhos de colônias. Os tamanhos de colônia de 10, 50 e 100 foram comparados em [49] . No entanto, a comparação em [49] foi baseado em um número igual de iterações (ciclos). Obviamente, com abelhas empregadas de tamanho 5 ABC gastará aproximadamente 10 avaliações de aptidão (abelhas empregadas, observadoras e uma possível abelha batedora) por iteração (ciclo), enquanto com abelhas empregadas de tamanho 50 ABC gastará aproximadamente 100 avaliações de aptidão por iteração (ciclo) . Portanto, a comparação baseada em iterações (ciclos) é pelo menos mais difícil de investigar, se for o caso. A partir dos gráficos de convergência apresentados em [49] (Figs. 7-11) a comparação entre diferentes tamanhos de colônias é difícil de seguir, pois as iterações (ciclos) devem ser mentalmente convertidas no número de avaliações de aptidão consumidas em cada iteração (ciclo). Uma tarefa que é quase impossível de fazer e conclusões erradas podem ser facilmente derivadas (Ver Seção 2) Portanto, nosso objetivo era replicar o experimento em [49] , mas desta vez a comparação foi baseada em igual número de avaliações de aptidão. Os resultados são apresentados em Figs. 1 e 2 (a mesma configuração experimental para f 1— f 5 foi usado como no experimento anterior com os resultados apresentados em tabela 1) Comparação dos gráficos de convergência de [49] (Deixou Figs. 1 e 2), que se baseiam em um número igual de iterações

(ciclos), com os gráficos de convergência (direita Figs. 1 e 2), que se baseiam em um número igual de avaliações de aptidão, revela

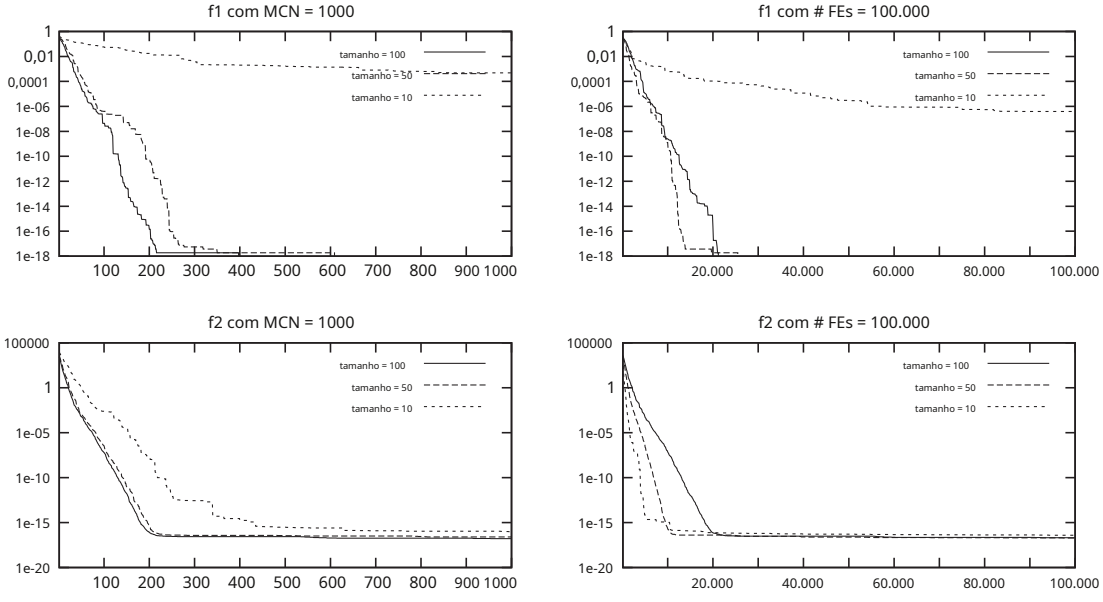

Figura 1. f 1 e f 2 com MCN = 1000 (esquerda) e # FEs = 100.000 (direita).

uma situação bem diferente. Pode ser visto nos gráficos de convergência (direita Figs. 1 e 2) que o tamanho da colônia mais apropriado depende muito do problema. Por exemplo, o mais econômico (solução boa e rápida) para f 2— f 3 é
tamanho da colônia ¼ 10, para f 1 e f 4 é tamanho da colônia ¼ 50, e tamanho da colônia ¼ 100 para f 5. Por outro lado, os gráficos de convergência (direita Figs. 1 e 2) revelar que tamanho da colônia ¼ 10 é uma configuração inadequada para f 1; f 4— f 5, e tamanho da colônia ¼ 50 é uma configuração inadequada para f 5, enquanto tamanho da colônia ¼ 100 para f 1— f 5 sempre encontram uma boa solução, embora a convergência não esteja entre as mais rápidas (por razões óbvias). Dos experimentos realizados em [49] os autores concluíram: " Da Tabela 4 e Figs. 7–12, pode-se concluir que conforme o tamanho da população aumenta, o algoritmo produz melhores resultados. No entanto, após um valor suficiente para o tamanho da colônia, qualquer incremento no valor não melhora o desempenho do algoritmo ABC significativamente. Para os problemas de teste realizados neste trabalho, o tamanho da colônia de 50–100 pode fornecer uma velocidade de convergência aceitável para pesquisa. " De acordo com os resultados apresentados nesta seção a citada conclusão do [49] só pode ser parcialmente suportado. A primeira parte da conclusão de [49] : " conforme o tamanho da população aumenta, o algoritmo produz melhores resultados " não foram apoiados por nossos experimentos. Parece que a 'regra de ouro': (uma população maior é necessária para problemas multimodais, de alta dimensão e de otimização mais difícil) freqüentemente usada em algoritmos meta-heurísticos também é válida para ABC. Enquanto a segunda parte da conclusão de [49] : " Para os problemas de teste realizados neste trabalho, o tamanho da colônia de 50-100 pode fornecer uma velocidade de convergência aceitável para pesquisa " ainda pode ser suportado. De fato, tamanho da colônia ¼ 100 para funções de benchmark incluídas f 1— f 5 é sempre uma escolha adequada, pois os resultados obtidos são de boa qualidade, mas nem sempre da convergência mais rápida (ver por exemplo f 3 dentro
Figura 2) Nossa descoberta de que o tamanho de colônia mais apropriado depende do problema não é surpreendente e está de acordo com a teoria e prática de algoritmos meta-heurísticos. Por exemplo, para a otimização de funções unimodais simples e de baixa dimensão, como a função de esfera f 2, um pequeno tamanho da população é suficiente e mais eficaz (f 2 pol
Figura 1) Nosso achado também está de acordo com o estudo recente [61] onde foi mostrado que, com parâmetros de controle ajustados corretamente (incluindo o tamanho da população), os resultados ABC originais podem ser melhorados. Este é um exemplo de que as conclusões tiradas nem sempre podem ser válidas se os algoritmos são comparados com base em um número igual de iterações, mas com um número diferente de avaliações de aptidão consumidas em cada iteração.

Os dois experimentos anteriores exibem erros diferentes cometidos durante a contagem das avaliações de aptidão. Uma questão óbvia é se um pequeno número de avaliações de aptidão extra também levaria a pequenas diferenças na qualidade da solução. O pequeno experimento a seguir mostra que esse não é o caso. A partir de f 4 (Figura 2 , tamanho da colônia ¼ 100) um gráfico de erro foi

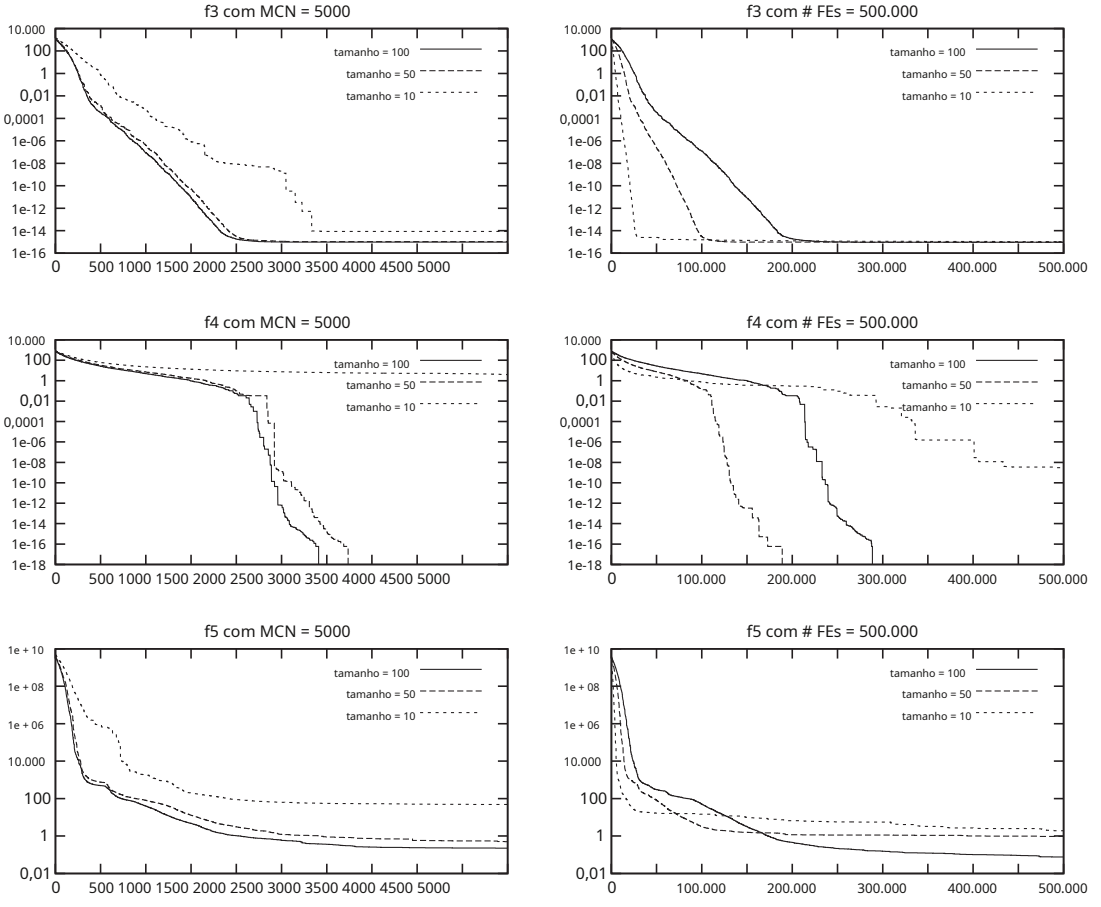

Figura 2. f 3, f 4 e f 5 com MCN = 5000 (esquerda) e # FEs = 500.000 (direita).

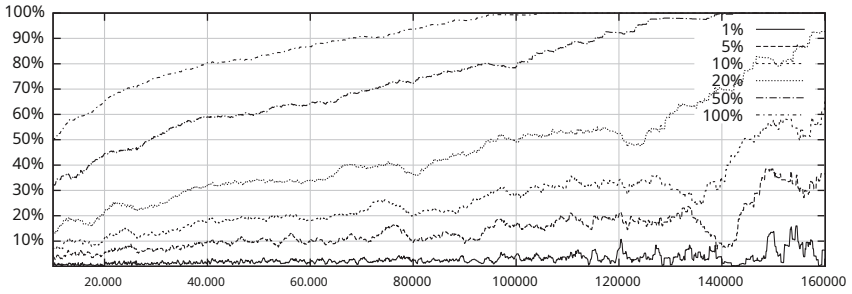

Fig. 3. Gráfico de erro para f 4 de Figura 2 e tamanho da colônia ¼ 100

gerado (ou seja, Fig. 3) mostrando o quão grande é um erro quando uma certa porcentagem de avaliações de aptidão extra são permitidas. Pode ser notado a partir de Fig. 3 que ao aumentar o número de avaliações de aptidão em 1%, 5%, 10%, 20%, 50% e 100%, o erro, ou melhora na aptidão, pode ser tão grande quanto 18%, 40%, 66%, 92 %, 100% e 100%, respectivamente. É claro que essas curvas dependem muito do problema e não podem ser generalizadas para outros problemas. Além disso, depende do estágio evolutivo. Durante os estágios iniciais, as melhorias por um pequeno número de avaliações de aptidão podem ser ótimas, mas em estágios posteriores, quando as melhorias não são mais alcançadas, as avaliações de aptidão apenas perdem tempo de processamento. Por este motivo, o gráfico em Fig. 3
mostra o estágio intermediário de evolução (de 10.000 a 160.000 avaliações de aptidão) para f 4 em relação ao gráfico de convergência de Figura 2 .

4. Conclusões

O principal objetivo deste estudo foi lembrar sobre vários equívocos ao comparar algoritmos meta-heurísticos com base em um número igual de iterações. Essa comparação é injusta se os algoritmos comparados não consomem o mesmo número de avaliações de aptidão por iteração. Foi demonstrado que tal prática é onipresente no caso das variantes ABC (Seção 2), visto que muitos pesquisadores negligenciaram as avaliações de aptidão consumidas durante a fase de abelhas escoteiras e durante a pesquisa local adicional (se introduzidas em variantes ABC), apesar de que no código ABC publicamente disponível era claramente observável que as abelhas escoteiras consumiam avaliações de aptidão.

A fim de superar o problema de comparação injusta com base em um número igual de iterações, o ABC foi reimplementado (ver Algoritmos 1 e 2 dentro Apêndice A) garantindo que o ABC não consumisse mais avaliações de aptidão do que o permitido. Como tal, o ABC pode ser facilmente usado para comparações justas com outros algoritmos no futuro, onde pode não haver ameaças de validade às conclusões devido a mais avaliações de aptidão consumidas do que o permitido. ABC é um algoritmo meta-heurístico de muito sucesso e muitas novas variantes ABC são previstas. Portanto, é muito importante que comparações justas com outros algoritmos sejam alcançadas e que quaisquer conclusões tiradas sejam apoiadas pelos experimentos realizados. Observe que o problema discutido neste artigo não é apenas pertinente para variantes ABC, mas problemas semelhantes também foram identificados em outros algoritmos meta-heurísticos. Por exemplo, na implementação original do TLBO, o número de avaliações de aptidão durante a etapa de eliminação duplicada foi apenas aproximado [81] , o que pode ser considerado impreciso e incorreto na contagem das avaliações de aptidão. Isso é especialmente verdadeiro quando uma aproximação foi definida para um determinado tamanho de população e, em seguida, usada para diferentes tamanhos de população (como em [103]) Além disso, os pesquisadores frequentemente relatam que as comparações entre diferentes algoritmos são inadequadas, uma vez que alguns algoritmos comparados consomem uma quantidade muito maior de tempo de CPU [72] . Este é frequentemente o caso, mas nem sempre quando a comparação é feita em um número igual de iterações, mas os algoritmos consomem muito mais avaliações de aptidão por iteração. No geral, é surpreendente como o problema discutido é onipresente. Por isso, chamamos atenção para a situação atual.

Apêndice A

A implementação do ABC básico está disponível gratuitamente em http://mf.erciyes.edu.tr/abc , que é uma razão adicional para a ABC popularidade. No entanto, essa implementação é baseada em um número igual de iterações (ciclos) e, portanto, menos adequada quando a comparação entre algoritmos meta-heurísticos é feita em um número igual de avaliações de aptidão. O pseudo-código do ABC usando contador para avaliações de aptidão e critério de terminação com base nas avaliações de aptidão máxima é apresentado em Algoritmo 1 . O pseudo-código do ABC usando critério de terminação baseado em MCN e SN, mas consumir uma avaliação de aptidão a menos durante a fase de abelha empregada para uma abelha empregada que se tornará uma escuteira, é apresentado em Algoritmo 2 . Observe que nesta implementação, a fase de abelha escuteira está entre a fase de abelha empregada e a fase de abelha observadora, enquanto é após a fase de abelha observadora no ABC básico ou na implementação 1. Também é possível classificar as fases em ordens diferentes . Isso mostra a flexibilidade do ABC.

Algoritmo 1. O pseudocódigo do ABC criança levada 1 ð FEs º.

```
Data: Set the control parameters of the ABC algorithm
SN: Number of Foods
limit: Maximum number of trial for abandoning a source
MFE: Maximum number of fitness evaluations
begin
    //Initialization;
    num_eval ⟵ 0 ;
    for s = 1 to SN do
        X(s) ⟵ random solution by Eq. 1 [3];
        f_s ⟵ f(X(s));
        trial(s) ⟵ 0;
        num_eval + + ;
    end
    repeat
        //Employed Bees Phase;
        for s = 1 to SN do
            x′ ⟵ a new solution produced by Eq. 2 [3];
            f(x′) ⟵ evaluate new solution;
            num_eval + +;
            if f(x′) < f_s then
                X(s) ⟵ x′; f_s ⟵ f(x′); trial(s) ⟵ 0;
            else
                trial(s) ← trial(s) + 1;
            end
            if num_eval == MFE then
                Memorize the best solution achieved so far and exit main repeat;
            end
        end
        Calculate the probability values p_i for the solutions using fitness values by Eqs. 3 and 4 [3];
        //Onlooker bee phase;
        s ⟵ 1; t ⟵ 1 ;
        repeat
            r ⟵ rand(0, 1);
            if r < p(s) then
                t ⟵ t + 1;
                x′ ⟵ a new solution produced by Eq. 2 [3];
                f(x′) ⟵ evaluate new solution;
                num_eval + +;
                if f(x′) < f_s then
                    X(s) ⟵ x′; f_s ⟵ f(x′); trial(s) ⟵ 0;
                else
                    trial(s) ⟵ trial(s) + 1;
                end
                if num_eval == MFE then
                    Memorize the best solution achieved so far and exit main repeat;
                end
            end
            s ⟵ (s mod SN) + 1;
        until t = SN ;
        //Scout Bee Phase;
        mi ⟵ {s : trial(s) = max(trial)};
        if trial(mi) >= limit then
            X(mi) ⟵ random solution by Eq. 1 [3];
            f_mi ⟵ f(X(mi));
            num_eval + + ;
            trial(mi) ⟵ 0;
            if num_eval == MFE then
                Memorize the best solution achieved so far and exit main repeat;
            end
        end
        Memorize the best solution achieved so far;
    until num_eval = MFE ;
end
```

Algoritmo 2. O pseudocódigo do ABC criança levada 2 ð FEs º.

```
Data: Set the control parameters of the ABC algorithm
SN: Number of Foods
limit: Maximum number of trial for abandoning a source
MCN: Maximum number of cycles
begin
    //Initialization;
    num_eval ⟵ 0 ;
    for s = 1 to SN do
        X(s) ⟵ random solution by Eq. 1 [3];
        f_s ⟵ f(X(s));
        trial(s) ⟵ 0;
        num_eval + +;
    end
    cycle ⟵ 1;
    while cycle < MCN do
        //Employed Bees Phase;
        mi ⟵ {s : trial(s) = max(trial)};
        for s = 1 to SN do
            if (trial(s) < limit or s ! = mi) then
                x′ ⟵ a new solution produced by Eq. 2 [3];
                f(x′) ⟵ evaluate new solution;
                num_eval + + ;
                if f(x′) < f_s then
                    X(s) ⟵ x′; f_s ⟵ f(x′); trial(s) ⟵ 0;
                else
                    trial(s) ⟵ trial(s) + 1;
                end
            end
        end
        Memorize the best solution achieved so far;
        //Scout Bee Phase;
        if (trial(mi) >= limit then
            X(mi) ⟵ random solution by Eq. 1 [3];
            f_mi ⟵ f(X(mi));
            num_eval + + ;
            trial(mi) ⟵ 0;
        end
        Calculate the probability values p_i for the solutions using fitness values by Eqs. 3 and 4 [3];
        //Onlooker Bees Phase;
        s ⟵ 1; t ⟵ 1 ;
        while t ≤ SN do
            r ⟵ rand(0, 1);
            if r < p(s) then
                t ⟵ t + 1;
                x′ ⟵ a new solution produced by Eq. 2 [3];
                f(x′) ⟵ evaluate new solution;
                num_eval + + ;
                if f(x′) < f_s then
                    X(s) ⟵ x′; f_s ⟵ f(x′); trial(s) ⟵ 0;
                else
                    trial(s) ⟵ trial(s) + 1;
                end
            end
            s ⟵ (s mod SN) + 1;
        end
        Memorize the best solution achieved so far;
        cycle + +;
    end
end
```

Referências

[1] MR Adaryani, A. Karami, Arti fi cial bee colony algoritmo para resolver o problema de fluxo de potência ideal multi-objetivo, Electr. Power Energy Syst. 53 (2013) 219–230 .

[2] A. Ahrari, M. Shariat-Panahi, AA Atai, GEM: um novo método de otimização evolucionária com busca de vizinhança aprimorada, Appl. Matemática. Comput. 210 (2009) 376-386 .

[3] B. Akay, D. Karaboga, um algoritmo modificado Arti fi cial Bee Colony para otimização de parâmetros reais, Inform. Sci. 192 (2012) 120-142 . [4] R. Akbari, A. Mohammadi, K. Ziarati, Um romance de algoritmo de otimização de enxame de abelhas para otimização de função numérica, Commun. Nonlin. Sci. Numer. Simul. 15 (2010) 3142-3155 .

[5] D. Aydin, S. Ozyon, C. Yasar, T. Liao, Algoritmo de colônia de abelhas arti fi cial com tamanho de população dinâmico para problema de despacho econômico e de emissão combinado, Electr. Power Energy Syst. 54 (2014) 144–153 .

[6] T. Bäck, DB Fogel, Z. Michalewicz, Handbook of Evolutionary Computations, Oxford University Press, 1996 . [7] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-to-far selection in Arti fi cial Bee Colony algorithm, Appl. Soft Comput. 11 (2011) 2888–2901 . [8] JC Bansal, H. Sharma, SS Jadon, Arti fi cial bee colony algoritmo: a survey, Int. J. Advan. Intell. Parad. 5 (2013) 123–159 . [9] RS Barr, BL Golden, JP Kelly, MGC Resende, WR Stewart Jr., Projetando e relatando experimentos computacionais com métodos heurísticos, J. Metaheur. 1 (1995) 9-32 .

[10] T. Bartz-Beielstein, Pesquisa Experimental em Computação Evolutiva: O Novo Experimentalismo, Springer, 2006 . [11] S. Biswas, A. Chatterjee, SK Goswami, Um algoritmo de mínimo quadrado de colônia de abelha artificial para resolver problemas de estimativa de harmônicos, Appl. Soft Comput. 13 (2013) 2343–2355 .

[12] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual compare, ACM Comput. Surv. 35 (2003) 268–308 . [13] AL Bolaji, AT Khader, MA Al-Betar, MA Awadallah, Algoritmo de colônia de abelha artificial, suas variantes e aplicações: uma pesquisa, J. Theoret. Appl. Informar. Technol. 47 (2013) 434-459 .

[14] K. Chandrasekaran, S. Hemamalini, SP Simon, NP Padhy, Compromisso de unidade térmica usando algoritmo de colônia de abelha artificial codificado binário / real, Electr. Power Syst. Res. 84 (2012) 109-119 .

[15] P.-T. Chang, J.-H. Lee, A Fuzzy DEA e modelo integrado de formulação de mochila para seleção de projetos, Comp. Oper. Res. 39 (2012) 112-125 . [16] S.-M. Chen, A. Sarosh, Y.-F. Dong, algoritmo de colônia de abelhas arti fi cial baseado em recozimento simulado para otimização numérica global, Appl. Matemática. Comput. 219 (2012) 3575–3589 .

[17] P. Civicioglu, Transformando coordenadas cartesianas geocêntricas em coordenadas geodésicas usando o algoritmo de busca diferencial, Comp. Geosci. 46 (2012) 229-247 .

[18] WG Cochran, Sampling Techniques, John Wiley & Sons, New York, NY, 1977 . [19] CC Columbus, SP Simon, Compromisso da unidade baseada no lucro: uma abordagem ABC paralela usando um cluster de estação de trabalho, Comp. Electr. Eng. 38 (2012) 724-745 . [20] M. Črepinšek, S.-H. Liu, M. Mernik, Exploração e exploração em algoritmos evolutivos: uma pesquisa, ACM Comput. Surv. 45 (2013) 35: 1-35: 33 . [21] M. Črepinšek, S.-H. Liu, M. Mernik, Replicação e comparação de experimentos computacionais em computação evolucionária aplicada: armadilhas comuns e diretrizes para evitá-los, Appl. Soft Comput. 19 (2014) 161-170 .

[22] G. Deng, Z. Xu, X. Gu, Um algoritmo discreto de colônia de abelhas arti fi cial para minimizar o tempo de fl uxo total na programação da loja de fl uxo de bloqueio, Chin. J. Chem. Eng. 20 (2012) 1067–1073 .

[23] J. Derrac, S. Garcia, D. Molina, F. Herrera, Um tutorial prático sobre o uso de testes estatísticos não paramétricos como uma metodologia para comparar algoritmos evolutivos e de inteligência de enxame, Swarm Evolution. Comput. 1 (1) (2011) 3-18 .

[24] AE Eiben, C. Schippers, Sobre a exploração e exploração evolucionária, Fund. Informar. 35 (1998) 35–50 .

[25] AE Eiben, M. Jelasity, Uma nota crítica sobre metodologia de pesquisa experimental na CE, em: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), 2002, pp. 582–587. [26] AE Eiben, JE Smith, Introdução à Computação Evolutiva, Springer, 2008 . [27] M. El-Abd, Avaliação de desempenho de algoritmos de forrageamento vs. algoritmos evolutivos, Inform. Sci. 182 (2012) 243-263 . [28] G. Fei, F. Feng-xia, D. Yan-fang, Q. Yi-bo, I. Balasingham, Um romance não-Lyapunov abordagem através do algoritmo de colônia de abelha artificial para detectar órbitas periódicas instáveis com ordens altas, Exp. Syst. Appl. 39 (2012) 12389–12397 .

[29] F. Gao, F.-X. Fei, Q. Xu, Y.-F. Deng, Y.-B. Qi, I. Balasingham, A novel arti fi cial bee colony algoritmo with space contraction for unknown parâmetros identi? Cation and time-delays of caotic systems, Appl. Matemática. Comput. 219 (2012) 552–568 .

[30] W. Gao, S. Liu, algoritmo de colônia de abelhas arti fi cial melhorado para otimização global, Inform. Processar. Lett. 111 (2011) 871-882 . [31] W. Gao, S. Liu, um algoritmo de colônia de abelhas arti fi cial modi fi cado, Comp. Oper. Res. 39 (2012) 687-697 . [32] W. Gao, S. Liu, L. Huang, A melhor algoritmo de colônia de abelhas arti fi cial global para otimização global, J. Comput. Appl. Matemática. 236 (2012) 2741–2753 . [33] S. Garcia, D. Molina, M. Lozano, F. Herrera, Um estudo sobre o uso de testes não paramétricos para analisar o comportamento de algoritmos evolutivos: um estudo de caso na sessão especial CEC'2005 sobre otimização de parâmetros reais, J Metaheur. 15 (6) (2009) 617-644 .

[34] H. Garg, M. Rani, SP Sharma, predição do comportamento incerto da unidade de impressão em uma indústria de papel usando colônia de abelhas arti fi cial e metodologia fuzzy Lambda-Tau, Appl. Soft Comput. 13 (2013) 1869-1881 .

[35] BANHEIRO. Hong, previsão de carga elétrica por SVR recorrente sazonal (regressão de vetor de suporte) com algoritmo de colônia de abelha artificial caótica, Energia 36 (2011) 5568–5578 .

[36] M.-H. Horng, Seleção de limiares multinível baseada no algoritmo de colônia de abelhas arti fi cial para segmentação de imagem, Exp. Syst. Appl. 38 (2011) 13785-13791 .

[37] T.-J. Hsieh, H.-F. Hsiao, W.-C. Sim, Prevendo mercados de ações usando transformadas wavelet e redes neurais recorrentes: um sistema integrado baseado em algoritmo de colônia de abelhas arti fi cial, Appl. Soft Comput. 11 (2011) 2510–2525 .

[38] T.-J. Hsieh, H.-F. Hsiao, W.-C. Sim, Dados de tendência de angústia financeira de mineração usando máquinas de vetores de suporte guiadas por penalidade baseadas em nhybrid de otimização de enxame de partículas e algoritmo de colônia de abelhas arti fi cial, Neurocomputing 82 (2012) 196–206 .

[39] T.-J. Hsieh, W.-C. Sim, Abelhas guiadas por penalidade procuram problemas de alocação de redundância com uma mistura de componentes em sistemas paralelos em série, Comp. Oper. Res. 39 (2012) 2688–2704 .

[40] C.-C. Hsu, H.-C. Chen, K.-K. Huang, Y.-M. Huang, um sistema de recomendação de material auxiliar personalizado baseado no estilo de aprendizagem no Facebook, aplicando um algoritmo de colônia de abelhas artificial, Comp. Matemática. Appl. 64 (2012) 1506–1513 .

[41] S.-J. Huang, X.-Z. Liu, Aplicação de otimização baseada em colônia de abelhas arti fi cial para estimativa de seção de falha em sistemas de energia, Electr. Power Energy Syst. 44 (2013) 210–218 .

[42] M. Husseinzadeh Kashan, N. Nahavandi, A. Husseinzadeh Kashan, DisABC: um novo algoritmo de colônia de abelhas arti fi cial para otimização binária, Appl. Soft Comput. 12 (2012) 342-352 .

[43] R. Irani, R. Nasimi, Aplicação de rede neural com base em colônia de abelhas arti fi cial na predição de pressão de fundo de poço em perfuração subequilibrada, J. Petrol. Sci. Eng. 78 (2011) 6–12 .

[44] HT Jadhav, R. Roy, algoritmo de colônia de abelhas arti fi cial guiada por Gbest para despacho ambiental / econômico considerando energia eólica, Exp. Syst. Appl. 40 (2013) 6385–6399 .

[45] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex arti fi cial bee colony algoritms, Comp. Struct. 87 (2009) 861-870 . [46] F. Kang, J. Li, Z. Ma, algoritmo de colônia de abelha artificial Rosenbrock para otimização global precisa de funções numéricas, Inform. Sci. 181 (2011) 3508-3531 .

[47] F. Kang, J. Li, H. Li, Algoritmo de colônia de abelha artificial e pesquisa de padrão hibridizado para otimização global, Appl. Soft Comput. 13 (2013) 1781–1791 .

[48] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization. Relatório Técnico-TR06, Universidade Erciyes, Faculdade de Engenharia, Departamento de Engenharia da Computação, 2005. [49] D. Karaboga, B. Basturk, On the performance of arti fi cial bee colony (ABC) algoritmo, Appl. Soft Comput. 8 (2008) 687-697 . [50] D. Karaboga, B. Akay, A comparative study of arti fi cial bee colony algorithm, Appl. Matemática. Comput. 214 (2009) 108-132 . [51] D. Karaboga, B. Akay, Uma pesquisa: algoritmos que simulam a inteligência do enxame de abelhas, Artif. Intell. Rev. 31 (2009) 61-85 . [52] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, Uma pesquisa abrangente: algoritmo e aplicações arti fi cial bee colony (ABC), Artif. Intell. Rev. 42 (2014) 21-57 .

[53] N. Karaboga, F. Latifoglu, Adaptive filtering barulhento transcranial Doppler sinal usando algoritmo de colônia de abelha artificial, Eng. Appl. Artif. Intell. 26 (2013) 677-684 .

[54] N. Karaboga, F. Latifoglu, Eliminação de ruído no sinal Doppler transcraniano usando filtros IIR projetados com colônia de abelhas arti fi cial - algoritmo ABC, Dig. Sig. Processar. 223 (2013) 1051–1058 .
[55] YS Kavian, A. Rashedi, A. Mahani, Z. Ghassemlooy, Roteamento e atribuição de comprimento de onda em redes ópticas usando algoritmo Arti fi cial Bee Colony, Optik 124 (2013) 1243–1249 .
[56] B. Kitchenham, P. Brereton, Uma revisão sistemática da pesquisa do processo de revisão sistemática em engenharia de software, Inform. Softw. Technol. 55 (2013) 2049–2075 .

[57] T. Krink, B. Filipič, GB Fogel, R. Thomsen, problemas de otimização ruidosa um desafio particular para evolução diferencial ?, em: Proceedings of 2004 Congress on Evolutionary Computation, 2004, pp. 332–339. [58] G. Li, P. Niu, X. Xiao, Desenvolvimento e investigação de algoritmo de colônia de abelhas arti fi cial eficiente para otimização de função numérica, Appl. Soft Comput. 12 (2012) 320-332 .

[59] Y. Li, Y. Wang, B. Li, um algoritmo de evolução diferencial assistida por colônia de abelhas arti fi cial híbrido para o fluxo de potência reativa ideal, Electr. Power Energy Syst. 52 (2013) 25–33 .
[60] X. Liao, J. Zhou, R. Zhang, Y. Zhang, Um algoritmo de colônia de abelhas arti fi cial adaptativo para despacho econômico de longo prazo em sistemas hidrelétricos em cascata, Electr. Power Energy Syst. 43 (2012) 1340–1345 .
[61] T. Liao, D. Aydin, T. Stützle, Colônias de abelhas arti fi ciais para otimização contínua: análise experimental e melhorias, Swarm Intell. 7 (2013) 327- 356 .

[62] M. Ma, J. Liang, M. Guo, Y. Fan, Y. Yin, segmentação de imagem SAR baseada em algoritmo de colônia de abelha artificial, Appl. Soft Comput. 11 (2011) 5205–5214 . [63] T. Malakar, SK Goswami, Despacho ativo e reativo com movimentos de controle mínimo, Electr. Power Energy Syst. 44 (2013) 78-87 . [64] VJ Manoj, E. Elias, Arti fi cial bee colony algoritmo para o projeto de transmultiplexador de banco de filtro não uniforme sem multiplicador, Inform. Sci. 192 (2012) 193–203 .

[65] M. Manuel, E. Elias, Projeto de filtro FIR de mascaramento de resposta de frequência no espaço de dígitos com sinais canônicos usando algoritmo de colônia de abelhas arti fi cial modi fi cado, Eng. Appl. Artif. Intell. 26 (2013) 660-668 .
[66] E. Mezura-Montes, O. Cetina-Dominguez, Análise empírica de uma colônia de abelhas arti fi cial modi fi cada para otimização numérica restrita, Appl. Matemática. Comput. 218 (2012) 10943–10973 .
[67] E. Miandoabchi, F. Daneshzand, WY Szeto, R. Zanjirani Farahani, Projeto de rede viária urbana discreta multi-objetivo, Comp. Oper. Res. 40 (2013) 2429–2449 .

[68] V. Mohanraj, M. Chandrasekaran, J. Senthilkumar, S. Arumugam, Y. Suresh, Sistema de recomendação on-line auto-adaptativo baseado em abordagem de forrageamento de abelhas conduzidas por ontologia, J. Syst. Softw. 85 (2012) 2439–2450 .
[69] E. Moreira Bernardino, A. Moreira Bernardino, JM Sanchez-Perez, JA Gomez-Pulido, MA Vega-Rodriguez, Resolvendo problemas de projeto de rede SONET em grande escala usando algoritmos inspirados em abelhas, Opt. Interruptor. Rede. 9 (2012) 97-117 .
[70] E. Moreira Bernardino, A. Moreira Bernardino, JM Sanchez-Perez, JA Gomez-Pulido, MA Vega-Rodriguez, Algoritmos de otimização de enxame aplicados a grandes redes de comunicação balanceadas, J. Netw. Comp. Appl. 36 (2013) 504–522 .
[71] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Relatório Técnico, Califórnia Institute of Technology, Concurrent Computation Program 158-79, 1989. [72] A. Mozaffari, M. Gorji-Bandpy, TB Gorji, Projeto ótimo de sistemas de engenharia de restrição: aplicação de algoritmo de abelha inteligente mutável, Int. J. Bio-Insp. Comput. 4 (2012) 167-180 .

[73] R. Mukherjee, S. Chakraborty, S. Samanta, Seleção de parâmetros de processo de usinagem de descarga elétrica de fio usando algoritmos de otimização não tradicionais, Appl. Soft Comput. 12 (2012) 2506–2516 .
[74] M. Nikolić, D. Teodorović, Estudo empírico do algoritmo Bee Colony Optimization (BCO), Exp. Syst. Appl. 40 (2013) 4609–4620 . [75] IMS de Oliveira, R. Schirru, Inteligência de enxame de abelhas arti fi ciais aplicada à otimização de gerenciamento de combustível In-core, Ann. Nucl. Energy 38 (2011) 1039–1045 .

[76] S. Ozyon, D. Aydin, Colônia de abelhas arti fi ciais incrementais com busca local para problema de despacho econômico com limites de taxa de rampa e zonas de operação proibidas, Energy Convers. Gerir. 65 (2013) 397–407 .
[77] Q.-K. Pan, MF Tasgetiren, PN Suganthan, TJ Chua, Um algoritmo de colônia de abelhas arti fi cial discreto para o problema de programação de loja de fluxo de lote, Inform. Sci. 181 (2011) 2455–2468 .
[78] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, estudos de mapeamento sistemático em engenharia de software, em: Proceedings of the 12th International Conferência sobre Avaliação e Avaliação em Engenharia de Software (EASE'08), 2008, pp. 71–80. [79] RV Rao, PJ Pawar, Otimização de parâmetros de um processo de fresagem multipass usando algoritmos de otimização não tradicionais, Appl. Soft Comput. 10 (2010) 445-456 .

[80] RV Rao, VK Patel, Otimização de torre de resfriamento úmido de contrafluxo de tiragem mecânica usando algoritmo de colônia de abelhas arti fi cial, Energy Convers. Gerir. 52 (2011) 2611-2622 .
[81] RV Rao, VK Patel, Um algoritmo de otimização baseado em ensino-aprendizagem elitista para resolver problemas complexos de otimização restrita, Int. J. Indust. Eng. 3 (2013) 535–560 .
[82] MM Rashidi, N. Galanis, F. Nazari, A. Basiri Parsa, L. Shamekhi, Análise paramétrica e otimização de ciclos regenerativos de Clausius e Rankine orgânicos com dois aquecedores de água de alimentação usando colônia de abelhas arti fi ciais e rede neural artificial, Energia 36 (2011) 5728 –5740 .
[83] FJ Rodriguez, M. Lozano, C. Garcia-Martinez, JD Gonzalez-Barrera, Um algoritmo de colônia de abelhas arti fi cial para o problema de agrupamento maximamente diverso, Inform. Sci. 230 (2013) 183–196 .
[84] S. Saadi, A. Guessoum, M. Bettayeb, modelo de rede neural otimizado ABC para desfoque de imagem com sua implementação FPGA, Microprocess. Microsyst. 37 (2013) 52-64 .

[85] SL Sabat, SK Udgata, A. Abraham, Algoritmo de colônia de abelha artificial para extração de parâmetro de modelo de pequeno sinal de MESFET, Eng. Appl. Artif. Intell. 23 (2010) 689-694 .
[86] O. Safarzadeh, A. Zolfaghari, A. Norouzi, H. Minuchehr, Otimização do padrão de carregamento de reatores PWR usando Arti fi cial Bee Colony, Ann. Nucl. Energy 38 (2011) 2218-2226 .
[87] S. Samanta, S. Chakraborty, Otimização paramétrica de alguns processos de usinagem não tradicionais usando algoritmo de colônia de abelhas arti fi cial, Eng. Appl. Artif. Intell. 24 (2011) 946-957 .
[88] A. Singh, An arti fi cial bee colony algorithm for the leaf-constrained Minimum Spanning Tree problem, Appl. Soft Comput. 9 (2009) 625-631 . [89] TSD Singh, A. Chatterjee, MMSE design of nonlinear Volterra equalizers using arti fi cial bee colony algoritmo, Measurement 46 (2013) 210–219 . [90] M. Sonmez, Arti fi cial bee colony algoritmo para otimização de estruturas de treliça, Appl. Soft Comput. 11 (2011) 2406–2418 . [91] R. Storn, K. Price, Evolução diferencial - uma heurística simples e eficiente para otimização global em espaços contínuos, J. Global Optim. 11 (1997) 341-359 .

[92] Z.-G. Su, P.-H. Wang, J. Shen, Y.-F. Zhang, L. Chen, modelo fuzzy Convenient T – S com desempenho aprimorado usando uma nova técnica de agrupamento fuzzy inteligente de enxame, J. Process Control 22 (2012) 108–124 .
[93] Z.-G. Su, P.-H. Wang, J. Shen, Y.-G. Li, Y.-F. Zhang, E.-J. Hu, Abordagem de particionamento fuzzy automático usando o algoritmo Arti fi cial Bee Colony (VABC) de comprimento de string variável, Appl. Soft Comput. 12 (2012) 3421–3441 .
[94] G. Sun, G. Li, Q. Li, substituto baseado em design de fi delidade variável e algoritmo de colônia de abelhas arti fi cial para o processo de conformação de chapas de metal, Finite Elem. Anal. Des. 59 (2012) 76–90 .
[95] H. Sun, H. Lus, R. Betti, Identificação de modelos estruturais usando um algoritmo Arti fi cial Bee Colony modi fi cado, Comp. Struct. 116 (2013) 59-74 . [96] S. Sundar, A. Singh, Uma abordagem de inteligência de enxame para o problema da árvore de abrangência mínima quadrática, Inform. Sci. 180 (2010) 3182-3191 . [97] WY Szeto, Y. Wu, SC Ho, um algoritmo de colônia de abelhas arti fi cial para o problema de roteamento de veículos capacitados, Euro. J. Operat. Res. 215 (2011) 126–135 . [98] AA Taleizadeh, STA Niaki, H.-M. Pequeno problema da cadeia de suprimentos de um único fornecedor e um único comprador com demanda estocástica e tempo de espera difuso, sistema baseado em conhecimento. 48 (2013) 1-9 .
[99] P. Tapkan, L. Ozbakir, A. Baykasoglu, Modeling and resolvendo problema de balanceamento de linha de montagem bilateral restrita via algoritmos de abelha, Appl. Soft Comput. 12 (2012) 3343–3355 .
[100] MF Tasgetiren, Q.-K. Pan, PN Suganthan, AH-L. Chen, Um algoritmo discreto de colônia de abelhas arti fi cial para a minimização de tempo de fl uxo total em lojas de fl uxo de permutação, Inform. Sci. 181 (2011) 3459–3475 .
[101] MF Tasgetiren, Q.-K. Pan, PN Suganthan, A. Oner, A discrete arti fi cial bee colony algoritmo para o problema de programação de fluxo de permutação não ociosa com o critério de atraso total, Appl. Matemática. Modelo. 37 (2013) 6758-6779 .
[102] J.-P. Tien, T.-HS Li, Hybrid Taguchi-chaos of multinível imune e o algoritmo de colônia de abelha artificial para identificação de parâmetros de sistemas caóticos, Comp. Matemática. Appl. 64 (2012) 1108–1119 .
[103] G. Waghmare, Comentários sobre uma nota sobre algoritmo de otimização baseado em ensino-aprendizagem, Inform. Sci. 229 (2013) 159-169 . [104] X. Wang, X. Xie, TCE Cheng, um algoritmo de colônia de abelhas arti fi cial modi fi cado para aceitação de pedido em lojas de fl uxo de duas máquinas, Int. J. Produto. Econ. 141 (2013) 14-23 .
[105] B. Wu, C. Qian, W. Ni, S. Fan, Busca de harmonia híbrida e algoritmo de colônia de abelha artificial para problemas de otimização global, Comp. Matemática. Appl. 64 (2012) 2621–2634 .
[106] Z. Xian, J. Xie, Y. Wang, Representative arti fi cial bee colony algoritms: a survey, in: Proceedings of 2nd International Conference on Logistics, Informatics and Service Science, 2012, pp. 1419–1424. [107] W.-L. Xiang, M.-Q. An, Um algoritmo de colônia de abelhas arti fi cial eficiente e robusto para otimização numérica, Comp. Oper. Res. 0 (2013) 1256–1265 . [108] C. Xu, H. Duan, Arti fi cial bee colony (ABC), abordagem de função potencial de borda otimizada (EPF) para reconhecimento de alvo para aeronaves de baixa altitude, Patt. Recog. Lett. 31 (2010) 1759–1772 .
[109] C. Xu, H. Duan, F. Liu, Chaotic arti fi cial bee colony approach to Unabited Combat Air Vehicle (UCAV) path planning, Aerosp. Sci. Technol. 14 (2010) 535-541 .
[110] X. Yan, Y. Zhu, W. Zou, L. Wang, Uma nova abordagem para agrupamento de dados usando algoritmo de colônia de abelha artificial híbrido, Neurocomputing 97 (2012) 241–250 . [111] BANHEIRO. Sim, T.-J. Hsieh, Resolvendo problemas de alocação de redundância de confiabilidade usando um algoritmo de colônia de abelhas artificial, Comp. Oper. Res. 38 (2011) 1465-1473 .
[112] AR Yildiz, Otimização de parâmetros de corte em torneamento multi-passe usando abordagem baseada em colônia de abelhas arti fi cial, Inform. Sci. 220 (2013) 399–407 . [113] C. Zhang, D. Ouyang, J. Ning, An arti fi cial bee colony approach for clustering, Exp. Syst. Appl. 37 (2010) 4761-4767 . [114] H. Zhang, Y. Zhu, W. Zou, X. Yan, Um algoritmo de colônia de abelha artificial multi-objetivo híbrido para otimização de carga da produção de tira de cobre, Appl. Matemática. Modelo. 36 (2012) 2578–2591 .
[115] Z. Zhang, J. Lin, Y. Shi, Aplicação do algoritmo de colônia de abelhas arti fi cial à máxima verossimilhança DOA Estimation, J. Bionic Eng. 10 (2013) 100–109 . [116] R. Zhang, S. Song, C. Wu, Um algoritmo de colônia de abelhas arti fi cial híbrido para o problema de programação de job shop, Int. J. Produto. Econ. 141 (2013) 167-178 . [117] W. Zhang, N. Wang, S. Yang, algoritmo de colônia de abelha artificial híbrido para estimativa de parâmetro de célula de combustível de membrana de troca de prótons, Int. J. Hydro. Energy 38 (2013) 5796–5806 .
[118] G. Zhu, S. Kwong, algoritmo de colônia de abelha artificial guiada por Gbest para otimização de função numérica, Appl. Matemática. Comput. 217 (2010) 3166-3173 . [119] K. Ziarati, R. Akbari, V. Zeighami, On the performance of bee algoritms for resource-restrito project scheduling problem, Appl. Soft Comput. 11 (2011) 3720-3733 .