Ciências da Informação 181 (2011) 4684–4698

Listas de conteúdos disponíveis em ScienceDirect

Ciências da Informação

página inicial da revista: www.elsevier. com / locate / ins

Algoritmo de otimização de acasalamento de abelhas para o problema do caixeiro viajante euclidiano

Yannis Marinakis uma , * , Madalena Marinaki b , Georgios Dounias c

uma Universidade Técnica de Creta, Departamento de Engenharia de Produção e Gestão, Laboratório de Sistemas de Apoio à Decisão, 73100 Chania, Grécia
b Universidade Técnica de Creta, Departamento de Engenharia de Produção e Gestão, Laboratório de Controle de Sistemas Industriais, 73100 Chania, Grécia
c Universidade do Egeu, Departamento de Engenharia Financeira e de Gestão, Laboratório de Engenharia de Gestão e Decisão, 31 Fostini Str., 82100 Chios, Grécia

informações do artigo

Historia do artigo:
Disponível online 1 de julho de 2010

Palavras-chave:
Metaheurísticas
Otimização do acasalamento das abelhas Problema do caixeiro viajante

abstrato

Este artigo apresenta uma nova abordagem híbrida algorítmica inspirada na natureza baseada na Otimização de Acasalamento das Abelhas Melíferas para resolver com sucesso o Problema do Caixeiro Viajante Euclidiano. O algoritmo proposto para a solução do problema do caixeiro viajante, o Honey Bees Mating Optimization (HBMOTSP), combina um algoritmo Honey Bee Mating Optimization (HBMO), o algoritmo Multiple Phase Neighborhood Search-Greedy Randomized Adaptive Search Procedure (MPNS-GRASP) e a Estratégia de Expansão de Busca de Vizinhança Além desses dois procedimentos, o algoritmo proposto possui, também, duas características inovadoras principais adicionais em comparação com outros algoritmos de Otimização de Acasalamento das Abelhas Melíferas no que diz respeito ao operador de crossover e às operárias. A principal contribuição deste trabalho é mostrar que o HBMO pode ser utilizado em síntese híbrida com outras metaheurísticas para a solução do TSP com resultados notáveis tanto em qualidade quanto em eficiência computacional. O algoritmo proposto foi testado em um conjunto de 74 instâncias de benchmark do TSPLIB e em todas, exceto onze instâncias, a solução mais conhecida foi encontrada. Para as demais instâncias, a qualidade da solução produzida desvia menos de 0,1% do ótimo.

1. Introdução

Diversos processos biológicos e naturais têm influenciado as metodologias em ciência e tecnologia de maneira crescente nos últimos anos. Processos de controle de feedback, neurônios artificiais, a descrição da molécula de DNA e questões genômicas semelhantes, estudos do comportamento de sistemas imunológicos naturais e mais, representam alguns dos domínios de muito sucesso desse tipo em uma variedade de aplicações do mundo real. Durante a última década, a inteligência inspirada na natureza tornou-se cada vez mais popular por meio do desenvolvimento e utilização de paradigmas inteligentes no projeto de sistemas de informação avançados. O pensamento baseado em equipe interdisciplinar tenta fertilizar a compreensão da engenharia e das ciências da vida em sistemas interoperáveis avançados. [1] Entre o

* Autor correspondente. Tel .: +30 2821 037282; fax: +30 2821 069410.
Endereço de e-mail: marinakis@ergasya.tuc.gr (Y. Marinakis), magda@dssl.tuc.gr (M. Marinaki), g.dounias@aegean.gr (G. Dounias).
[1] Ação de coordenação FP-6 da UE NISIS: Nature Inspired Smart Information Systems (FP6–2002-IST-C) (http://www.nisis.de)

0020-0255 / $ - consulte o assunto
doi: 10.1016 / j.ins.2010.06.032

as abordagens mais populares inspiradas na natureza, quando a tarefa é a otimização dentro de domínios complexos de dados ou informações, são aqueles métodos que representam o comportamento de equipes de animais e microrganismos bem-sucedidos, tais como:

Inteligência de enxame ou bando: bandos de pássaros ou cardumes de peixes inspiraram a otimização de enxame de partículas [42] , Sistemas imunológicos artificiais: esquemas de algoritmos que imitam as características dos sistemas imunológicos biológicos [14,15] , Colônias de formigas: o comportamento de forrageamento das formigas deu origem à Otimização da Colônia de Formigas [18] etc.

Uma série de ferramentas inspiradas na natureza têm sido usadas para resolver problemas muito diversos de gestão da cadeia de suprimentos e operações, como programação, organização da produção e problemas de roteamento de veículos [68] .

Nos últimos anos, vários algoritmos de inteligência de enxame, baseados no comportamento das abelhas, foram apresentados. [9] . Esses algoritmos são divididos principalmente, em duas categorias de acordo com seu comportamento na natureza, o comportamento de forrageamento e o comportamento de acasalamento. As abordagens mais importantes que simulam o comportamento de forrageamento das abelhas são o Algoritmo Arti fi cial Bee Colony (ABC) proposto por Karaboga e Basturk. [40,41] , o Virtual Bee Algorithm proposto por Yang [84] , o algoritmo de otimização da colônia de abelhas proposto por Teodorovic e DellOrco [77] , o algoritmo BeeHive proposto porWedde et al. [83] , o Bee SwarmOptimization Algorithm proposto por Drias et al. [19] e o Algoritmo das Abelhas proposto por Pham et al. [63] .

O algoritmo Arti fi cial Bee Colony [40,41] é, principalmente, aplicado em problemas de otimização contínua e simula o comportamento de dança balançada que um enxame de abelhas realiza durante o processo de forrageamento das abelhas. Neste algoritmo, existem três grupos de abelhas:

(1) As abelhas empregadas, ou seja, as abelhas que determinam a fonte de alimento (soluções possíveis) a partir de um conjunto pré-especificado de fontes de alimento e compartilhar esta informação (dança de abanar) com as outras abelhas da colmeia,
(2) As abelhas observadoras, ou seja, as abelhas que, com base nas informações que recebem das abelhas empregadas, buscam um melhor fonte de alimento na vizinhança das fontes de alimento memorizadas, e
(3) As abelhas escoteiras, isto é, abelhas empregadas cuja fonte de alimento foi abandonada e uma nova busca de fonte de alimento começou (aleatoriamente).

O algoritmo de abelha virtual [84] é, também, aplicado em problemas de otimização contínua. Nesse algoritmo, a população de abelhas é associada a uma memória, a uma fonte de alimento e, então, todas as memórias se comunicam entre elas por meio de um procedimento de dança balançante. Todo o procedimento é semelhante com um algoritmo genético e foi aplicado em dois problemas de otimização de função com dois parâmetros. Na colmeia [83] algoritmo, um protocolo inspirado na linguagem da dança e comportamento de forrageamento das abelhas melíferas. Otimização do Enxame de Abelhas [19] , inicialmente a abelha encontra uma solução inicial (fonte de alimento) e a partir dessa solução as demais soluções são produzidas com o uso de determinadas estratégias. Então, cada abelha é atribuída a uma solução e quando as abelhas realizam sua busca, elas se comunicam entre si com uma estratégia de dança balançante e a melhor solução se tornará a nova solução de referência. Para evitar o ciclo, os autores usam uma lista tabu. No Algoritmo das Abelhas [63] , uma população de soluções iniciais (fontes de alimentos) são geradas aleatoriamente. Em seguida, as abelhas são atribuídas às soluções com base em sua função de aptidão. As abelhas voltam para a colmeia e, com base em suas fontes de alimento, várias abelhas são designadas à mesma fonte de alimento para encontrar uma solução de vizinhança melhor. Otimização da Colônia de Abelhas [77] algoritmo, uma solução passo a passo é produzida por cada abelha forrageira e quando as forrageadoras retornam à colmeia, uma dança de balanço é realizada por cada forrageira. Em seguida, as outras abelhas, com base em uma probabilidade, seguem as forrageadoras. Este algoritmo se parece com o Ant Colony Optimization [18] algoritmo, mas não usa o conceito de trilhas de feromônio.

Ao contrário do fato de existirem muitos algoritmos que se baseiam no comportamento de forrageamento das abelhas, o principal algoritmo proposto com base no comportamento de casamento é o algoritmo de Otimização de Acasalamento das Abelhas Melíferas (HBMO), que foi apresentado em [1,2] . Desde então, tem sido usado em vários aplicativos diferentes [3,21,33,76] . O algoritmo de otimização de acasalamento das abelhas melíferas simula o processo de acasalamento da rainha da colmeia. O processo de acasalamento da rainha começa quando a rainha voa para longe do ninho realizando o vôo de acasalamento durante o qual os zangões seguem a rainha e acasalam com ela no ar [1,3] . O algoritmo é um algoritmo de inteligência de enxame, pois usa um enxame de abelhas onde existem três tipos de abelhas: a rainha, os zangões e as operárias. Existem vários procedimentos que podem ser aplicados dentro do enxame. No algoritmo de otimização de acasalamento das abelhas melíferas, é descrito o procedimento de acasalamento da rainha com os zangões. Primeiro, a rainha está voando aleatoriamente no ar e, com base em sua velocidade e energia, se ela encontrar um zangão, então existe a possibilidade de acasalar com ele. Mesmo se a rainha acasalar com o zangão, ela não cria diretamente uma ninhada, mas armazena o genótipo do zangão em sua espermateca e a ninhada é criada apenas quando o vôo de acasalamento foi concluído. Com o termo genótipo, queremos dizer algumas das características básicas dos drones, ou seja, parte da solução. Um operador de cruzamento é usado para criar as ninhadas. Em uma colmeia, o papel das operárias é simplesmente o cuidado da ninhada (ou seja, alimentá-las com a '' geléia real ”) e, portanto, elas são apenas uma fase de busca local no algoritmo de Otimização de Acasalamento das Abelhas. Assim, este algoritmo combina o processo de acasalamento da rainha e uma parte do comportamento de forrageamento das abelhas dentro da colmeia. Se uma ninhada é melhor (mais apta) do que a rainha, então essa ninhada substitui a rainha.

Neste artigo, como não existem métodos inspirados na natureza competitiva baseados na Otimização do Acasalamento das Abelhas Melíferas para a solução do Problema do Caixeiro Viajante (TSP), pelo menos até onde sabemos, decidimos desenvolver tal algoritmo

e testar sua eficiência em comparação com outros algoritmos metaheurísticos clássicos e inspirados na natureza. O algoritmo proposto adota as características básicas do algoritmo de otimização de acasalamento das abelhas melíferas inicialmente proposto. [1-3,21,33] e, também, faz um uso combinado de uma série de procedimentos diferentes em cada uma das subfases do algoritmo principal, a fim de aumentar a eficiência do algoritmo proposto. Mais especificamente, o algoritmo proposto usa:

O procedimento de pesquisa adaptativa aleatória gananciosa de pesquisa de vizinhança de várias fases (MPNS-GRASP) [56] para o cálculo da população inicial de abelhas e da rainha inicial. Este procedimento é usado para ter uma rainha mais competitiva.

A Pesquisa de Vizinhança em Expansão (ENS) [54] como estratégia de busca local para ter trabalhadores mais eficazes e diferenciados. Através da ENS, cada ninhada tem a possibilidade de selecionar aleatoriamente o número de operárias (fases de busca local) que serão utilizadas para a melhoria da sua solução.
Um novo operador de crossover baseado em um procedimento de memória adaptativa [70] e em um operador de cruzamento uniforme para ter ninhadas mais aptas. Este operador de cruzamento combina o genótipo da rainha e de mais de um zangão para produzir uma ninhada. A razão pela qual esse operador de cruzamento é usado é porque na vida real a rainha armazena em sua espermateca após o acasalamento do genótipo de todos os zangões e depois de retornar à colmeia ela produz as crias. O procedimento de memória adaptativa é usado para dar a possibilidade à rainha de armazenar de drones bons previamente selecionados (em voos de acasalamento anteriores) parte de suas soluções, para ser capaz de usá-los em um novo voo de acasalamento e para produzir ninhadas mais aptas.

O algoritmo proposto foi aplicado no mais clássico problema de otimização combinatória, o Problema do Caixeiro Viajante, com resultados notáveis, pois na maioria das instâncias utilizadas, o algoritmo encontrou a solução mais conhecida. A combinação de procedimentos proposta reduz, significativamente, o tempo computacional do algoritmo tornando o algoritmo mais rápido e eficiente e, portanto, adequado para resolver problemas de grande escala em curto tempo computacional. O restante do artigo está organizado da seguinte forma: Na próxima seção, é apresentada uma descrição do Problema do Caixeiro Viajante. Na terceira seção, o algoritmo proposto, o algoritmo Honey Bees Mating Optimization para o Problema do Caixeiro Viajante (HBMOTSP), é apresentado e analisado em detalhes.

2. O problema do caixeiro viajante

O Problema do Caixeiro Viajante (TSP) é o problema de encontrar o passeio mais curto por todas as cidades que um vendedor precisa visitar. O TSP é provavelmente o problema mais famoso e amplamente estudado no campo da otimização combinatória [32,45] . O TSP pertence à classe de problemas de otimização NP-hard [36] . Algoritmos para resolver o TSP podem ser divididos em duas classes, algoritmos exatos e algoritmos heurísticos. Os algoritmos exatos têm a garantia de encontrar a solução ótima em um número exponencial de etapas. Os algoritmos exatos mais eficazes são algoritmos de ramificação e corte [44] com o qual grandes instâncias de TSP foram resolvidas [5] . O problema com esses algoritmos é que eles são bastante complexos e exigem muito poder de computador [35] . As tentativas heurísticas para resolver o problema do caixeiro viajante são focadas em métodos de construção de viagens [10,67,72] e métodos de melhoria de turismo [24,48,49] . Os métodos de construção do tour criam uma solução passo a passo, enquanto os métodos de melhoria do tour começam com um tour inicial e, em seguida, tentam transformá-lo em um tour mais curto [37] .

Desde a introdução do Simulated Annealing [43] e Tabu Search [26,27] um avanço foi obtido com a introdução de metaheurísticas [29] que têm a possibilidade de encontrar sua saída dos ótimos locais. Para a solução do TSP, uma série de algoritmos metaheurísticos foram aplicados, como Simulated Annealing [11,62,67] , Pesquisa Tabu [25,28,30,85] , Algorítmos genéticos [8,51,55,61,64–66,71,78,80,81] , Pesquisa de Vizinhança Variável [58] , Pesquisa Local Iterada [5,37,60] , Redes neurais [4,7,57,59,74] e Greedy Randomized Adaptive Search Procedure (GRASP) [54] . Finalmente, nos últimos quinze anos, uma série de Nature Inspired ou Swarm Intelligence Methods, como Ant Colony Optimization [6,12,16-18,20,46,

50,53,75,79] e Otimização de Enxame de Partículas [31,47,52,73,82] foram propostas para a solução do TSP. Alguns anos atrás, um desafio de implementação foi organizado por Johnson e McGeoch [38] . Neste desafio, foi feito um esforço para coletar todos os algoritmos possíveis que foram publicados na área do Problema do Caixeiro Viajante. Além disso, esse desafio deu aos pesquisadores a oportunidade de apresentar, testar e comparar seus métodos usando as instâncias de referência da área. Os resultados do desafio foram apresentados em dois artigos [38,39] e na página web do desafio (http: //www.research. att.com/ dsj / chtsp /)

3. O algoritmo de otimização do acasalamento das abelhas

3.1. Descrição geral do algoritmo

O algoritmo proposto, o Algoritmo de otimização de acasalamento de abelhas melíferas para o problema do caixeiro viajante (HBMOTSP), combina vários procedimentos diferentes. Cada um deles corresponde a uma fase diferente do acasalamento

processo das abelhas. Inicialmente, temos que escolher a população de abelhas que irão con fi gurar a colméia inicial. Existem duas maneiras diferentes de calcular a população inicial, seja completamente ao acaso (como no algoritmo proposto inicialmente [1]) ou usando um algoritmo para obter as melhores soluções iniciais possíveis e, assim, obter uma rainha mais eficiente. No algoritmo proposto, a população inicial é criada usando uma versão modificada do Greedy Randomized Adaptive Search Procedure (GRASP) [22,69] , a Pesquisa de Vizinhança de Fase Múltipla-GRASP (MPNSGRASP) [54,56] . O algoritmo MPNS-GRASP foi usado em outro algoritmo híbrido proposto por Marinakis et al.

[56] e os resultados obtidos foram significativamente melhores do que os resultados obtidos quando uma população inicial aleatória foi criada. Esta é a razão pela qual usamos este método para a criação da solução inicial do problema atual. O melhor membro da população inicial de abelhas é selecionado como a rainha da colmeia. Todos os outros membros da população são os drones.

Antes de o processo de acasalamento começar, o usuário deve definir um número que corresponda ao tamanho da espermateca da rainha. Este número corresponde ao número máximo de acasalamentos da rainha em um único vôo de acasalamento. Cada vez que a rainha acasala com sucesso com um zangão, o genótipo do zangão é armazenado e uma variável é aumentada em um até que o tamanho da espermateca seja alcançado [2] . Outros dois parâmetros devem ser definidos, o número de rainhas e o número de ninhadas que nascerão por todas as rainhas. Nesta implementação do algoritmo Honey Bees Mating Optimization (HBMO), o número de rainhas é definido igual a um, pois na vida real apenas uma rainha sobreviverá em uma colmeia, e o número de ninhadas é definido igual ao número correspondente ao tamanho da espermateca da rainha. Então, o vôo de acasalamento da rainha começa. No início do vôo, a rainha é inicializada com algum conteúdo de energia (inicialmente, a velocidade e a energia da rainha são geradas aleatoriamente) e retorna ao seu ninho quando a energia é inferior a um valor limite (thres) e a espermateca não está cheia [3] . Um zangão acasala com uma rainha probabilisticamente usando a seguinte função de recozimento [1,2] :

$$\text{Prob}(D) = e^{\frac{-D(f)}{\text{Velocidade}(t)}} \qquad (1)$$

Onde Prob (D) é a probabilidade de adicionar o esperma do drone D para a espermateca da rainha (ou seja, a probabilidade de um acasalamento bem-sucedido), D (f) é a diferença absoluta entre a aptidão de D e a aptidão da rainha (para uma descrição completa do cálculo da função de aptidão veja abaixo) e Velocidade (t) é a velocidade da rainha na hora t. A probabilidade de acasalamento é alta quando a rainha ainda está no início de seu vôo de acasalamento, portanto, sua velocidade é alta, ou quando a aptidão do zangão é tão boa quanto a aptidão da rainha. Após cada transição no espaço, a velocidade e a energia da rainha diminuem de acordo com as seguintes equações:

$$\text{Velocidade}(t+1) = \text{uma}\ \text{Velocidade}(t) \qquad (2)$$

$$\text{energia}(t+1) = \text{uma}\ \text{energia}(t) \qquad (3)$$

Onde uma é um fator 2 (0,1) que determina a quantidade que a velocidade e a energia serão reduzidas após cada transição e cada etapa. Deve-se observar que a Eq. (3) é diferente do proposto por Abbass [1,2] e foi introduzido desta forma porque gostaríamos de correlacionar diretamente a redução da velocidade com a redução da energia e, também, usar menos parâmetros. Inicialmente, a velocidade e a energia da rainha são geradas aleatoriamente. Uma série de voos de acasalamento são realizados. No início de um voo de acasalamento, drones são gerados aleatoriamente e a rainha seleciona um drone usando a regra probabilística na Eq. (1) . Se o acasalamento for bem-sucedido (ou seja, o zangão passa na regra de decisão probabilística), o esperma do zangão é armazenado na espermateca da rainha.

Usando um operador cruzado, uma nova ninhada (solução teste) é formada, a qual mais tarde pode ser melhorada empregando trabalhadores para conduzir buscas locais. Um novo operador de crossover é desenvolvido para simular o procedimento que ocorre na vida real, onde a rainha armazena vários espermatozóides de diferentes zangões em sua espermateca e usa partes do genótipo dos diferentes zangões para criar a nova ninhada. Assim, a qualidade da nova solução (a ninhada) é a mais adequada porque, como aceita partes de diferentes soluções (rainha e zangões), ela tem mais habilidades de exploração. Esta é uma grande diferença do algoritmo proposto em comparação com outros algoritmos de otimização de acasalamento das abelhas. [1,3,21,33,76] e aos algoritmos evolutivos clássicos.

Na vida real, o papel das operárias é restrito ao cuidado da ninhada e por esta razão as operárias não são membros separados da população, mas são utilizadas como procedimentos de busca local a fim de melhorar as ninhadas produzidas pelo vôo de acasalamento da rainha. Cada um dos trabalhadores tem capacidades diferentes e a escolha de dois trabalhadores diferentes pode produzir soluções diferentes. Isso é realizado com o uso de uma série de heurísticas de pesquisa local simples e suas combinações. Assim, um dos
parâmetros do algoritmo é o número de heurísticas de pesquisa local (C_1) e o outro é o número de suas combinações
(C_2). A soma desses dois números ($w = w_1 + C_2$) dá o número de trabalhadores. Cada uma das crias escolherá, aleatoriamente, uma operária para alimentá-la com geleia real (fase de busca local) tendo como resultado a possibilidade de substituir a rainha caso a solução de
a ninhada é melhor do que a solução da rainha atual. Se a ninhada não conseguir substituir a rainha, então, no próximo vôo de acasalamento da rainha, essa ninhada será um dos zangões se tiver uma função de aptidão melhor do que um dos zangões atuais. O número de drones permanece constante em cada iteração e a população de drones consiste nos drones mais aptos de todos

gerações. Um pseudocódigo do algoritmo proposto é apresentado a seguir, enquanto nas próximas seções os procedimentos do algoritmo são explicados em detalhes.

Algoritmo 1: Otimização de acasalamento de abelhas melíferas para TSP

```
Inicialização
Gere a população inicial de abelhas usando MPNS-GRASP Seleção da melhor
abelha como a rainha
Seleção do número máximo de voos de acasalamento ( M) Fase
Principal
fazer enquanto eu 6 M
    Inicialize a espermateca da rainha, energia e velocidade. Selecione uma

    fazer enquanto energia> três e a espermateca não está cheia
        Selecione um drone
        E se o drone passa na condição probabilística então
            Adicionar esperma do zangão na espermateca
        fim se
    Velocidade (t + 1) = uma Velocidade (t)
    energia (t + 1) = uma energia (t)
    enddo
    Faz j = 1, Tamanho da Espermateca
        Selecione um esperma da espermateca
        Gere uma ninhada aplicando um operador de cruzamento entre a rainha,
            os drones selecionados e a memória adaptativa Selecione,
        aleatoriamente, um trabalhador
        Use o trabalhador selecionado para melhorar a aptidão da ninhada (estratégia ENS)
        E se a aptidão da ninhada é melhor do que a aptidão da rainha então
            Substituir a rainha com a ninhada
          senão
            E se a aptidão da ninhada é melhor do que a aptidão do zangão então substitua o
            zangão com a ninhada
            fim se
          fim se
    enddo
enddo
Retorna The Queen (melhor solução encontrada)
```

3.2. População inicial

APERTO [22,54,69] é um método iterativo de pesquisa em duas fases. Cada iteração consiste em duas fases, uma fase de construção e um procedimento de pesquisa local. Na fase de construção, um função gananciosa aleatória é usado para construir uma solução inicial. Essa técnica aleatória fornece uma solução viável em cada iteração. Esta fase pode ser descrita como um processo que adiciona gradativamente um elemento de cada vez a uma solução parcial (incompleta). A escolha do próximo elemento a ser adicionado é determinada ordenando todos os elementos em uma lista de candidatos, o Lista de candidatos restritos (RCL), com respeito a uma função gananciosa. O componente probabilístico de um APERTO é caracterizado por escolher aleatoriamente um dos melhores candidatos da lista, mas não necessariamente o candidato principal. O algoritmo guloso é um procedimento simples, de uma passagem, para resolver o Problema do Caixeiro Viajante. Na segunda fase, um busca local é inicializado a partir desses pontos e o resultado final é simplesmente a melhor solução encontrada em todas as pesquisas.

As diferenças mais importantes de MPNS-GRASP do GRASP clássico diz respeito à construção da lista RCL e à aplicação de funções gulosas alternativas em cada iteração ao invés de apenas uma função gulosa simples como na abordagem clássica. Além disso, no MPNS-GRASP uma combinação de funções gulosas também é possível. O algoritmo começa com uma função gulosa e se os resultados não forem melhorando uma função gulosa alternativa é usada em seu lugar (um limite valor b 2 determina este procedimento) [56] . A utilização de uma busca local simples na segunda fase do algoritmo clássico limita as chances de se obter melhores soluções, portanto, o MPNS-GRASP utiliza a Busca por Vizinhança em Expansão, que é uma estratégia de busca local muito fl exível (ver Seção 3,5) [54] . O pseudocódigo do algoritmo MPNS-GRASP é fornecido abaixo.

Algoritmo 2: MPNS-GRASP

```
Inicialização
Selecione o conjunto de algoritmos que serão utilizados na Fase Principal 1 do MPNS-GRASP Selecione o conjunto de
algoritmos de busca local que serão utilizados na Fase Principal 2 do MPNS-GRASP
Selecione o valor limite b 2
! b 2 é a qualidade de aceitação da solução usando um método de construção de tour específico em! Fase 1
principal do algoritmo após um número pré-especificado de iterações
Selecione o tamanho inicial do RCL
Fase principal
Defina o número de iterações igual a zero
fazer enquanto número máximo de iterações não foi alcançado
   Fase Principal 1
      Aumente o contador de iteração
      Construa o RCL
      Selecione aleatoriamente a partir do RCL o elemento candidato para inclusão na solução parcial
         Ligar um algoritmo ganancioso
      E se o número de iterações é igual ao número pré-especificado então
            E se a qualidade da melhor solução para a fase principal 1 é maior do que o limite b 2 então
                  Escolha outra heurística de construção
            fim se
      fim se
   Fase Principal 2
            Ligar Expansão da pesquisa de vizinhança
enddo
Retorna A melhor solução
```

3,3. Cálculo da função de aptidão

No TSP, a aptidão de cada indivíduo está relacionada ao comprimento da rota de cada círculo. Uma vez que os problemas com os quais lidamos são problemas de minimização, se uma solução viável tem um grande valor de função objetivo, ela é caracterizada como uma candidata a solução não promissora e, portanto, sua adequação deve ser definida como um valor pequeno. Inversamente, um grande valor de aptidão deve corresponder a uma solução com um baixo valor de função objetivo. Uma forma de fazer isso é encontrar inicialmente o indivíduo na população com o valor máximo da função objetivo e subtrair desse valor o valor da função objetivo de cada um dos outros indivíduos. Ao fazer isso, o maior valor de aptidão corresponde ao passeio com o menor comprimento. Como a probabilidade de selecionar um indivíduo para o acasalamento está relacionada à sua aptidão e como o indivíduo com o pior valor de função objetivo tem aptidão igual a zero, ele nunca será selecionado para o acasalamento. Portanto, para evitar sua exclusão total, a aptidão de todos os indivíduos desta população é incrementada em um, resultando, assim, em uma pior aptidão individual.

3.4. Operador de crossover

Propomos um operador de cruzamento que inicialmente identifica as características comuns dos indivíduos pais e, em seguida, as copia para as ninhadas. Este operador de crossover é um tipo de procedimento de memória adaptativa. Inicialmente, a memória adaptativa foi proposta por Rochat e Taillard [70] como parte de uma metaheurística Tabu Search para a solução do problema de roteamento de veículos. Este procedimento armazena características (caminhos no Problema do Caixeiro Viajante) de boas soluções. Cada vez que uma nova boa solução é encontrada, a memória adaptativa é atualizada. Em nosso caso, na primeira geração a memória adaptativa está vazia. Para adicionar uma solução ou parte de uma solução na memória adaptativa, existem três possibilidades:

(1) O candidato para a solução de memória adaptativa é uma melhor solução anterior (rainha) que tem o valor da função de aptidão em a maioria 10% pior do que o valor da melhor solução atual.
(2) O candidato para a solução de memória adaptativa é um membro da população (drone) que tem o valor da função de aptidão no máximo 10% pior do que o valor da melhor solução atual. (3) Um caminho é comum para a rainha e para vários zangões.

Mais analiticamente, neste operador de crossover, os pontos são selecionados aleatoriamente a partir da memória adaptativa, a partir do drones selecionados e da rainha. Assim, inicialmente dois números de operador de crossover são selecionados (Cr 1 e Cr 2) que controlam a fração dos parâmetros selecionados para a memória adaptativa, os drones selecionados e a rainha. Se houver comum partes nas soluções (rainha, zangões e memória adaptativa), então essas partes comuns são herdadas da ninhada, caso contrário, o Cr 1 e Cr 2 os valores são comparados com a saída de um gerador de números aleatórios, rand eu(0,1). Se o número aleatório for menor ou igual ao Cr 1 o valor correspondente é herdado da rainha, se o número aleatório estiver entre os Cr 1 e a

Cr 2 então o valor correspondente é herdado, aleatoriamente, de uma das soluções de elite que estão na memória adaptativa, caso contrário, é selecionado, aleatoriamente, das soluções dos zangões que estão armazenadas na espermateca. Assim, se a solução do rainha é denotada por q eu(t) (t é o número da iteração), a solução na memória adaptativa é denotada por de Anúncios eu(t) e a solução do drone por d eu(t), então, a solução da ninhada b eu(t) É dado por:

$$b_{eu}(t+1)=\begin{cases} q_{eu}(t); & \text{E se } rand_{eu}(0;1)\le Cr_1 \\ de\ Anúncios_{eu}(t); & \text{E se } Cr_1<rand_{eu}(0;1)\le Cr_2 \\ d_{eu}(t); & \text{por outro lado:} \end{cases} \qquad (4)$$

A cada iteração, a memória adaptativa é atualizada com base na melhor solução.

3,5. Trabalhadores - expandindo a pesquisa de vizinhança

Como já foi mencionado, as operárias não são membros separados da população, mas são usadas como procedimentos de busca local para melhorar as ninhadas produzidas pelo vôo de acasalamento da rainha. O método de pesquisa local usado neste artigo é a Pesquisa de Vizinhança em Expansão [54] . Expanding Neighborhood Search (ENS) é um algoritmo metaheurístico [54] que pode ser usado para a solução de uma série de problemas de otimização combinatória com resultados notáveis. As principais características desse algoritmo são (a) o uso da Estratégia de Movimentos de Busca Local Restritos por Círculo, (b) a habilidade do algoritmo de mudar entre diferentes estratégias de busca local e (c) o uso de uma estratégia de expansão. Esses recursos são explicados em detalhes a seguir.

No Círculo de movimentos de pesquisa local restritos - CRLSM estratégia, o tempo computacional é diminuído significativamente em comparação com outros algoritmos heurísticos e metaheurísticos porque todas as arestas que não vão melhorar a solução são excluídas do procedimento de busca. Isso acontece ao restringir a pesquisa em círculos ao redor das arestas candidatas à exclusão. Estratégia de movimentos de busca local restrita por círculo para um Movimento de teste de 2 opções [48] foi apresentado. Existem três possibilidades com base nos custos dos candidatos para exclusão e bordas de inclusão:

Se ambas as novas bordas aumentarem no custo, um movimento de teste de 2 opções não pode reduzir o custo do passeio (por exemplo, em Figura 1 (Possibilidade A), para ambas as novas arestas os custos C 2 e C 4 são maiores do que os custos B 2 e UMA de ambas as bordas antigas).
Se uma das duas novas arestas tiver um custo maior do que a soma dos custos das duas arestas antigas, um movimento de teste de 2 opções, novamente, não pode reduzir o custo do passeio (por exemplo, em Figura 1 (Possibilidade B), o custo da nova aresta C 3 é maior que a soma dos custos A + B 3 das arestas antigas).

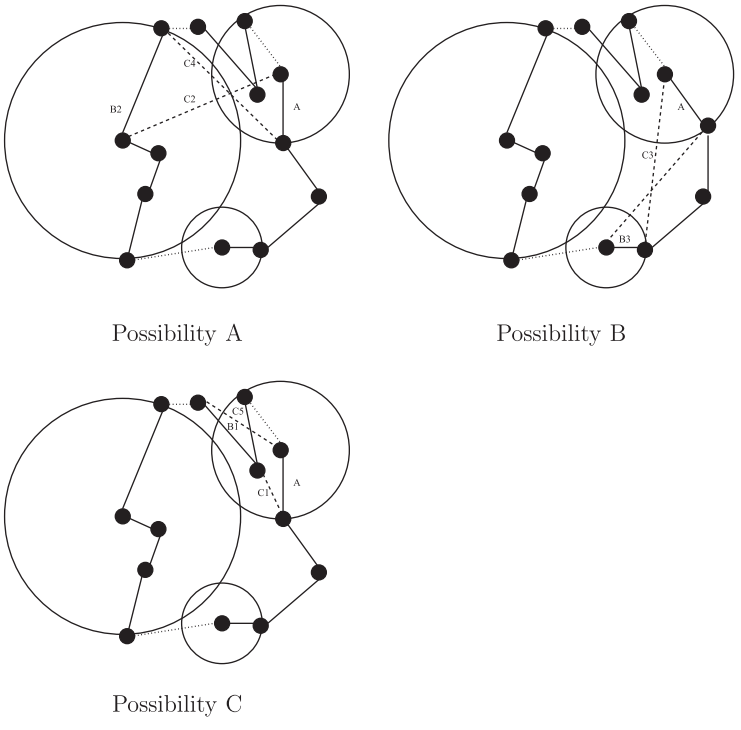

Figura 1. Explicação da estratégia de movimentos de busca local restrita por círculo para um movimento de teste de 2 opções.

O único caso em que uma mudança de teste de 2 opções pode reduzir o custo do passeio é quando pelo menos uma nova aresta custou menos do que o custo de uma das duas arestas antigas (por exemplo, em Figura 1 (Possibilidade C), o custo C 1 da nova aresta é menor que o custo da aresta antiga A) e a outra aresta custou menos do que a soma dos custos das duas arestas antigas (por exemplo, C 5 < A + B 1 in Figura 1 (Possibilidade C)).

Levando essas observações em consideração, a estratégia de Movimentos de Busca Local Restritos por Círculo restringe a busca às arestas onde um de seus nós finais está dentro de um círculo com comprimento do raio no máximo igual à soma dos custos (comprimentos) dos dois candidatos para exclusão arestas.

O algoritmo ENS tem a capacidade de alternar entre diferentes estratégias de busca local. A ideia de usar um bairro maior para escapar de um mínimo local para um melhor foi proposta inicialmente por Garfinkel e Nemhauser [23] e recentemente por Hansen e Mladenovic [34] . Gar fi nkel e Nemhauser propuseram uma maneira muito simples de usar uma vizinhança maior. Em geral, se com o uso de um bairro um ótimo local for encontrado, então um bairro maior é usado na tentativa de escapar do ótimo local. Hansen e Mladenovic propuseram um método mais sistemático para mudar entre diferentes bairros, denominado Pesquisa de Vizinhança Variável.

Por outro lado, o método ENS começa com um comprimento pré-especificado do raio do círculo da estratégia CRLSM. Dentro desse círculo, diferentes estratégias de busca local são aplicadas até que todos os movimentos de teste possíveis tenham sido explorados e a solução não possa ser melhorada nesta vizinhança. Posteriormente, o comprimento do raio do círculo é aumentado e, novamente, o mesmo procedimento é repetido até que o critério de parada seja ativado. As principais diferenças do ENS em relação aos outros dois métodos é o uso da Estratégia de Movimento de Busca Local Restrita por Círculo, que restringe a busca em círculos ao redor dos candidatos para bordas de exclusão e a maneira mais sofisticada que a estratégia de busca local pode ser alterada dentro dos círculos.

Na estratégia ENS, o tamanho da vizinhança é expandido em cada iteração externa [54] . Cada comprimento diferente do vizinhança constitui uma iteração externa. Inicialmente, o tamanho do bairro, f, é definido com base na estratégia de movimentos de busca local restritos por círculo, por exemplo f = A / 2, onde UMA é o custo de um dos candidatos para bordas de exclusão. Para o tamanho selecionado da vizinhança, uma série de diferentes estratégias de busca local são aplicadas até que todos os movimentos de teste possíveis tenham sido explorados e a solução não possa ser melhorada nesta vizinhança. As estratégias de busca local são alteradas com base em duas condições, primeiro se a estratégia de busca local atual encontra um ótimo local e segundo se a qualidade de a solução atual permanece maior do que o número limite b 1 para uma série de iterações internas. (A qualidade é dada em termos do desvio relativo da solução mais conhecida). Se a qualidade da solução for inferior ao número limite e o algoritmo para, caso contrário, a vizinhança é expandida aumentando o comprimento do raio da estratégia CRLSM f por uma porcentagem h (h é determinado empiricamente após investigação completa e é definido como igual a 10%) e o algoritmo continua. Quando o comprimento do raio da estratégia CRLSM é igual a UMA, o comprimento continua a aumentar até que o comprimento se torne igual a A + B, Onde B é o comprimento do outro candidato para borda de exclusão. Se o comprimento do raio da estratégia CRLSM for igual a A + B, e o algoritmo atingiu o número máximo de iterações, então o algoritmo termina com a solução atual. Dentro Figura 2 , o Método de Pesquisa de Vizinhança em Expansão é apresentado. O pseudocódigo a seguir descreve essa abordagem.

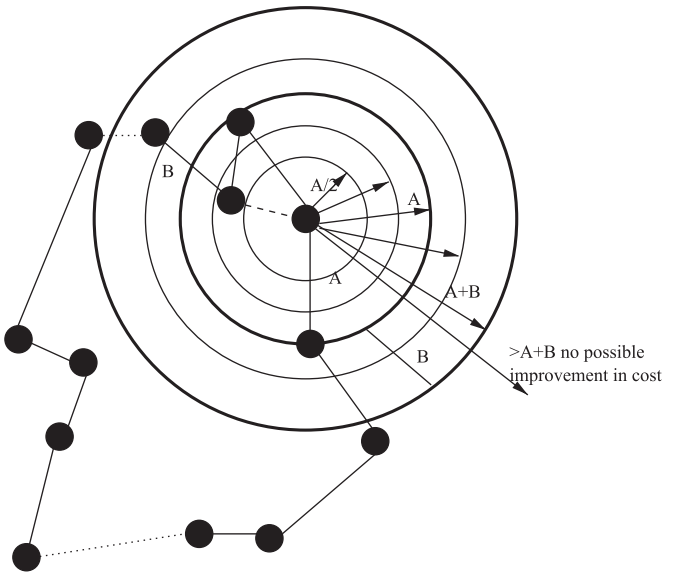

Figura 2. Explicação da estratégia de busca de vizinhança em expansão para um movimento experimental de 2 opções.

Algoritmo 3: Expansão da pesquisa de vizinhança

```
f = A / 2, m = 0! m = índice para os métodos de pesquisa locais
fazer enquanto ( x> e ou f <A + B)! x = a qualidade da solução
    m = 1
    Ligar primeira estratégia de pesquisa local
    E se x < e então
      PARE com a solução atual
    senão
      fazer enquanto ( m <M 1)
      ! M 1 = o número de estratégias de busca local
          E se x < b 1 ou um ótimo local é encontrado então
              Mudar método de busca local
              m = m + 1
              E se x < e então
                  PARE com a solução atual
              fim se
          fim se
      enddo
    fim se
  f = 1,1 f
  Atualizar b 1 e e
enddo
```

Na estratégia de Pesquisa de Vizinhança em Expansão, três algoritmos de pesquisa local são usados. O 2-opt, a 2,5-opt e a 3-opt algoritmos que foram introduzidos por Lin [48] . No primeiro algoritmo, a função de vizinhança é definida como a troca de duas arestas da solução atual com duas outras arestas que não estão na solução atual, no segundo, exceto pela troca de 2 arestas, há adicionalmente um movimento de inserção de nó, enquanto no terceiro método a função de vizinhança é definida como a troca de três arestas da solução atual por três outras arestas que não estão na solução atual.

4. Resultados computacionais

Toda a abordagem algorítmica foi implementada em Fortran 90 e foi compilada usando o compilador Lahey f95 em um Centrino Mobile Intel Pentium M 750 a 1,86 GHz, executando Suse Linux 9.1. Os parâmetros do algoritmo proposto foram selecionados após testes completos. Uma série de valores alternativos diferentes foram testados e os selecionados são aqueles que deram os melhores resultados computacionais no que diz respeito à qualidade da solução e ao tempo computacional necessário para alcançar esta solução. Assim, os parâmetros selecionados são: o número de rainhas é definido igual a 1, o número de drones é definido igual a 200, o número de voos de acasalamento (M) é definido igual a 1000, o tamanho da espermateca da rainha é definido igual a 50, o número de ninhadas é definido igual a 50, o parâmetro uma é definido igual a 0,9, o número de trabalhadores diferentes (C) é definido igual a 7 ($C_1 = 3$, $C_2 = 4$), o tamanho do RCL é definido igual a 50 e, finalmente, o valor limite (thres) é definido igual a 10^{10} Deve-se notar que a seleção dos parâmetros usados no algoritmo foi realizada levando em consideração uma série de questões:

A razão pela qual o número de rainhas é igual a um é que na vida real na colméia apenas uma rainha existe.
Como na vida real a rainha acasala com vários zangões e não com todos eles, o tamanho da espermateca deve ser menor que o número de zangões. Levando isso em consideração, sempre mantemos o tamanho da espermateca igual a 25% do número de zangões.

Na vida real, em cada geração (vôo de acasalamento), o número de ninhadas pode ser diferente. No entanto, como gostaríamos de ter um algoritmo eficaz, escolhemos o número de crias como constante e igual ao tamanho da espermateca em cada voo de acasalamento. Uma série de valores no intervalo [10.5000] foi testada para o número de drones. Para esses valores, o tamanho da espermateca foi alterado em conformidade. Para um grande número de drones, o algoritmo precisava de um tempo computacional significativo sem dar uma melhoria nas soluções, enquanto quando o número de drones era pequeno, o tamanho correspondente da espermateca era ainda menor, levando a uma pequena diversidade de soluções.

O número de voos de acasalamento foi definido igual a 1000, pois para um número maior de voos de acasalamento os resultados não foram melhorados. O parâmetro uma foi definido como 0,9, pois gostaríamos de ter uma diminuição lenta da velocidade da rainha para realizar mais acasalamentos com os zangões.

Após a seleção dos parâmetros finais, 50 execuções diferentes com os parâmetros selecionados foram realizadas para cada uma das instâncias de benchmark.

tabela 1
Resultados do HBMOTSP.

Instância	HBMO-TSP	Ótimo	x uma v (%)	x (%)	CPU (s)	Instância	HBMO-TSP	Ótimo	x uma v (%)	x (%)	CPU (s)
Eil51	426	426	0	0	0,17	Rat575	6773	6773	0	0	57,18
Berlin52	7542	7542	0	0	0,19	P654	34643	34643	0	0	61,23
Eil76	538	538	0	0	0,53	D657	48912	48912	0	0	63,27
Pr76	108159	108159	0	0	0,53	Rat783	8806	8806	0	0	71,12
Rat99	1211	1211	0	0	0,58	dsj1000	18660556	18659688	0,012	0,0046	80,29
KroA100	21282	21282	0	0	0,62	Pr1002	259045	259045	0,001	0	80,57
KroB100	22141	22141	0	0	0,65	u1060	224094	224094	0	0	80,68
KroC100	20749	20749	0	0	0,63	vm1084	239297	239297	0,005	0	85,21
KroD100	21294	21294	0	0	0,64	Pcb1173	56892	56892	0,003	0	89,28
KroE100	22068	22068	0	0	0,61	D1291	50801	50801	0	0	91,14
Rd100	7910	7910	0	0	0,71	Rl1304	252948	252948	0	0	103,29
Eil101	629	629	0	0	0,87	Rl1323	270199	270199	0	0	113,43
Lin105	14379	14379	0	0	0,98	nrw1379	56638	56638	0,009	0	121,89
Pr107	44303	44303	0	0	1.01	Fl1400	20127	20127	0,011	0	198,67
Pr124	59030	59030	0	0	1.08	u1432	152270	152270	0,016	0	200,01
Bier127	118282	118282	0	0	1,11	Fl1577	22249	22249	0,022	0	227,28
Ch130	6110	6110	0	0	1,28	d1655	62149	62128	0,122	0,0338	241,67
Pr136	96772	96772	0	0	1,35	vm1748	336712	336556	0,189	0,0463	257,81
Pr144	58537	58537	0	0	1,68	u1817	57201	57201	0,028	0	289,12
Ch150	6528	6528	0	0	2.01	Rl1889	316536	316536	0,017	0	291,57
KroA150	26524	26524	0	0	2,12	D2103	80450	80450	0,041	0	350,78
Pr152	73682	73682	0	0	2,21	u2152	64267	64253	0,39	0,0217	357,23
Rat195	2323	2323	0	0	3,45	u2319	234256	234256	0,028	0	391,08
D198	15780	15780	0	0	4,27	Pr2392	378032	378032	0,026	0	401,28
KroA200	29368	29368	0	0	5.01	pcb3038	137694	137694	0,002	0	457,29
KroB200	29437	29437	0	0	4,99	fl 3795	28783	28772	0,37	0,0382	461,81
Ts225	126643	126643	0	0	5,38	fnl4461	182612	182566	0,35	0,0251	498,01
Pr226	80369	80369	0	0	5,41	rl5915	565530	565530	0,012	0	557,87
Gil262	2378	2378	0	0	6,58	rl5934	556080	556045	0,0127	0,0062	561,21
Pr264	49135	49135	0	0	8,21	pla7397	23261400	23260728	0,0086	0,0028	780,31
A280	2579	2579	0	0	8,5	rl11849	923288	923288	0,098	0	890,05
Pr299	48191	48191	0	0	8,67	usa13509	19982859	19982859	0,087	0	897,09
Rd400	15281	15281	0	0	19,57	brd14051	469388	469388	0	0	902,35
Fl417	11861	11861	0	0	24,67	d15112	1573084	1573084	0,005	0	928,34
Pr439	107217	107217	0	0	36,21	d18512	645501	645244	0,25	0,039	995,17
Pcb442	50778	50778	0	0	37,12	pla33810	66079424	66050535	0,18	0,0437	1095,45
D493	35002	35002	0	0	45,21	pla85900	142495128	142383704	0,21	0,0782	1198,21

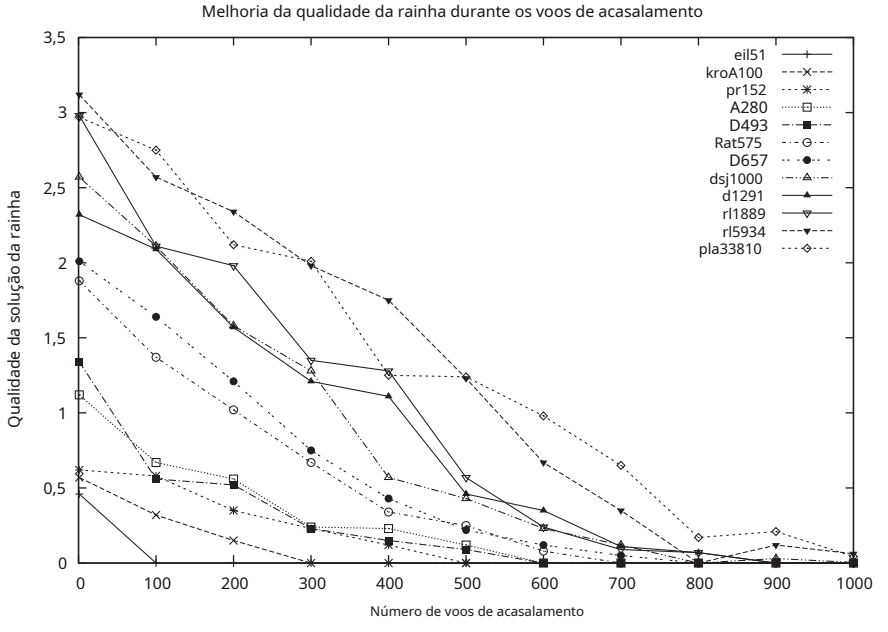

Fig. 3. Melhoria da qualidade da rainha em vários casos diferentes durante os voos de acasalamento.

O algoritmo foi testado em um conjunto de 74 problemas de amostra euclidiana com tamanhos variando de 51 a 85900 nós. Cada instância é descrita por seu nome e tamanho TSPLIB, por exemplo, em tabela 1 a instância chamada Pr76 tem tamanho igual a 76 nós. Dentro tabela 1 , a primeira coluna mostra o nome da instância (os números nos nomes denotam os nós de cada instância), a segunda coluna mostra a solução do HBMOTSP, a terceira coluna dá a solução ótima para cada instância tirada do
TSPLIB, a quarta coluna mostra a qualidade média das 50 execuções do algoritmo (X uma v), a quinta coluna mostra a qualidade da melhor execução do algoritmo proposto (X) e, finalmente, a sexta coluna dá o tempo computacional (em segundos), do
melhor execução, necessária para encontrar a solução. A qualidade [32] das soluções produzidas é dado em termos do desvio relativo
do ótimo, isto é X ¼ 100 ð c HBMOTSP c optar °, Onde c HBMOTSP denota o custo do ótimo encontrado pelo HBMOTSP, e c optar
é o custo da solução ótima.

Como pode ser observado nesta Tabela, a solução ótima foi encontrada em todos os casos, exceto onze, ou seja, em 85,3% de todos os casos. Para as outras instâncias, a qualidade é inferior a 0,1%. Além disso, nesta tabela o tempo computacional necessário para encontrar a melhor solução pelo HBMOTSP é apresentado. O tempo de CPU necessário é baixo para as instâncias com número de nós menor que 1000. Quando o número de nós é maior que 1000, o tempo de CPU é de alguma forma aumentado, mas ainda é muito eficiente. Ressalta-se que o objetivo deste estudo foi apresentar um algoritmo muito rápido e eficaz e, assim, a escolha dos parâmetros (como o número de drones ou o número de gerações) foi realizada de forma a que o algoritmo para combinar uma convergência rápida com os melhores resultados possíveis. Esse problema em 11 de 74 casos foi o motivo pelo qual o algoritmo não encontrou o ótimo. Se aumentarmos o número de drones e o número de gerações, o algoritmo melhora ainda mais as soluções, mas então o algoritmo encontra essas soluções em um tempo computacional superior. A diferença na qualidade dos resultados da melhor corrida em relação à qualidade média das 50 corridas está entre 0,00% e 0,3683%, enquanto a diferença média é igual a 0,029%. Em todas as 50 execuções do algoritmo, a solução mais conhecida foi encontrada em 46 instâncias. Para as outras instâncias, apenas em uma ou duas execuções o algoritmo não encontrou o ótimo. Porém, mesmo que a melhor solução não fosse encontrada em todas as execuções, as soluções encontradas estavam muito próximas das melhores soluções. Se aumentarmos o número de drones e o número de gerações, o algoritmo melhora ainda mais as soluções, mas então o algoritmo encontra essas soluções em um tempo computacional maior. A diferença na qualidade dos resultados da melhor corrida em relação à qualidade média das 50 corridas está entre 0,00% e 0,3683%, enquanto a diferença média é igual a 0,029%. Em todas as 50 execuções do algoritmo, a solução mais conhecida foi encontrada em 46 instâncias. Para as outras instâncias, apenas em uma ou duas execuções o algoritmo não encontrou o ótimo. Porém, mesmo que a melhor solução não fosse encontrada em todas as execuções, as soluções encontradas estavam muito próximas das melhores soluções. Se aumentarmos o número de drones e o número de gerações, o algoritmo melhora ainda mais as soluções, mas então o algoritmo encontra essas soluções em um tempo computacional maior. A diferença na qualidade dos resultados da melhor corrida em relação à qualidade média das 50 corridas está entre 0,00% e 0,3683%, enquanto a diferença média é igual a 0,029%. Em todas as 50 execuções do algoritmo, a solução mais conhecida foi encontrada em 46 instâncias. Para as outras instâncias, apenas em uma ou duas execuções o algoritmo não encontrou o ótimo. Porém, mesmo que a melhor solução não fosse encontrada em todas as execuções, as soluções encontradas estavam muito próximas das melhores soluções. A diferença na qualidade dos resultados da melhor corrida em relação à qualidade m

mesa 2
Comparações do algoritmo proposto com outros algoritmos (o método 1 corresponde a ENS-GRASP, o método 2 corresponde a MPNS-GRASP1, o método 3 corresponde a MPNS-GRASP2, o método 4 corresponde a HybGEN e o método 5 corresponde a HBMOTSP).

Instância	Métodos					Instância	Métodos				
	1	2	3	4	5		1	2	3	4	5
Eil51	0	0	0	0	0	Rat575	1,32	0,23	0	0,68	0
Berlin52	0	0	0	0	0	P654	0,18	0,12	0	0,10	0
Eil76	0	0	0	0	0	D657	1,26	0,53	0	0,48	0
Pr76	0	0	0	0	0	Rat783	1.03	0,54	0	0,37	0
Rat99	0	0	0	0	0	dsj1000	1,17	0,28	0,0081	0,21	0,0046
KroA100	0	0	0	0	0	Pr1002	1,16	0,02	0	1,16	0
KroB100	0	0	0	0	0	u1060	1.01	0,44	0	0,27	0
KroC100	0	0	0	0	0	vm1084	1,18	0,34	0	0,51	0
KroD100	0	0	0	0	0	Pcb1173	1,37	0,04	0	1,57	0
KroE100	0	0	0	0	0	D1291	1,60	0,22	0	1,60	0
Rd100	0	0	0	0	0	Rl1304	0,88	0,40	0	0,88	0
Eil101	0	0	0	0	0	Rl1323	1.07	0,39	0	1.06	0
Lin105	0	0	0	0	0	nrw1379	1,21	0,33	0	0,82	0
Pr107	0	0	0	0	0	Fl1400	0,90	0,54	0	0,90	0
Pr124	0	0	0	0	0	u1432	1.07	0,66	0	0,71	0
Bier127	0,03	0	0	0	0	Fl1577	0,80	0,32	0	0,97	0
Ch130	0	0	0	0	0	d1655	1,21	0,67	0,0466	0,71	0,0338
Pr136	0	0	0	0	0	vm1748	1,25	0,62	0,0588	0,68	0,0463
Pr144	0	0	0	0	0	u1817	1,21	0,18	0	0,35	0
Ch150	0	0	0	0	0	Rl1889	0,85	0,23	0	0,85	0
KroA150	0	0	0	0	0	D2103	1.07	0,11	0	1.07	0
Pr152	0	0	0	0	0	u2152	1,12	0,48	0,0544	0,99	0,0217
Rat195	0,34	0	0	0	0	u2319	0,99	0,58	0	0,52	0
D198	0,05	0	0	0,04	0	Pr2392	2,11	0,29	0	1,62	0
KroA200	0,04	0	0	0,04	0	pcb3038	2,27	0,24	0	0,89	0
KroB200	0,15	0	0	0	0	fl 3795	2,35	0,41	0,059	0,99	0,0382
Ts225	0	0	0	0	0	fnl4461	2,57	0,61	0,0597	1.02	0,0251
Pr226	0,05	0	0	0	0	rl5915	2,89	0,10	0	0,85	0
Gil262	0,29	0	0	0	0	rl5934	2,61	0,37	0,0071	0,87	0,0062
Pr264	0	0	0	0	0	pla7397	2,31	0,69	0,0033	0,75	0,0028
A280	0,38	0	0	0,04	0	rl11849	3,01	0,68	0	1.02	0
Pr299	0,09	0,09	0	0,09	0	usa13509	3,15	0,06	0	1.08	0
Rd400	0,68	0,68	0	0,35	0	brd14051	3,21	0,65	0	0,47	0
Fl417	0,28	0,28	0	0,24	0	d15112	2,95	0,57	0	0,34	0
Pr439	0,17	0,17	0	0,17	0	d18512	2,21	0,23	0,043	0,28	0,039
Pcb442	0,33	0,33	0	0,32	0	pla33810	3,01	0,20	0,0469	0,77	0,0437
D493	0,71	0,63	0	0,57	0	pla85900	3,29	0,73	0,0993	0,61	0,0782

Dentro Fig. 3 , a melhoria da qualidade da rainha durante diferentes voos de acasalamento é apresentada. O valor da qualidade é dado para cada 100 voos de acasalamento. São apresentados os resultados de doze instâncias diferentes, sete com número de cidades menor que 1000 e cinco com número de cidades maior que 1000. Como pode ser observado para o exemplo eil51, a qualidade da rainha tornou-se igual ao ótimo em apenas 100 voos de acasalamento. Para as outras instâncias com pequeno número de cidades, a solução ótima foi encontrada em mais voos de acasalamento, mas sempre em menos de 700 voos de acasalamento. Nas instâncias com um grande número de nós, as qualidades iniciais foram piores do que as qualidades das outras instâncias e, portanto, mais voos de acasalamento foram necessários para encontrar o ótimo.

A fim de dar a eficiência do algoritmo proposto e o papel de cada componente deste algoritmo, as comparações são realizadas com outros algoritmos metaheurísticos (mesa 2), nomeadamente com o Expanding Neighborhood Search-GRASP (ENS-GRASP) proposto por Marinakis et al. [54] , com duas versões do Multiple Phase Neighbourhood Search-GRASP (MPNS-GRASP1 e MPNS-GRASP2) proposto por Marinakis et al. [56] e com o Hybrid Genetic-GRASP (HybGEN) proposto por Marinakis et al. [55] . No ENS-GRASP, a fase de pesquisa local do GRASP é a Pesquisa de Vizinhança em Expansão, conforme descrito na Seção 3,5 . O MPNS-GRASP1 e MPNS-GRASP2 usam mais procedimentos de busca local do que os usados nesta implementação do MPNS-GRASP. Eles usam uma variante do conhecido algoritmo Lin-Kernighan chamado Random Backtracking Lin-Kernighan (RBLK). O MPNS-GRASP2, também, usa um Path Relinking Process (PR). A razão pela qual esses dois procedimentos não foram usados no algoritmo HBMOTSP é que gostaríamos de ter uma versão mais simples do MPNS-GRASP para enfatizar mais nas características do algoritmo de otimização de acasalamento das abelhas melíferas. O MPNS-GRASP aplicado neste artigo não é complicado e é muito fácil de ser implementado. O algoritmo HybGEN é um algoritmo Hybrid Genetic onde o ENS-GRASP é usado na fase de mutação de um algoritmo genético simples para melhorar as soluções. A razão pela qual o algoritmo HybGEN é usado nas comparações é porque gostaríamos de mostrar o fato de que o algoritmo HBMOTSP fornece resultados superiores em comparação com outro algoritmo baseado em população. Outra razão pela qual usamos o HybGEN é porque neste algoritmo foi proposto pela primeira vez o operador de crossover que usamos no HBMOTSP. Nós podemos ver de mesa 2 que os resultados do algoritmo proposto são melhores a partir dos resultados dos três (ENS-GRASP, MPNS-GRASP1 e HybGEN) dos quatro algoritmos. Os resultados do algoritmo MPNS-GRASP2 são quase iguais aos do algoritmo proposto, ou seja, ambos os algoritmos encontram nas mesmas instâncias o ótimo, mas nas onze instâncias em que ambos os algoritmos não encontraram o ótimo, os resultados encontrados pelo HBMOTSP são melhores em todas as instâncias do MPNS-GRASP2. Esta é uma conclusão muito importante das comparações porque quando foi decidido o uso de MPNS-GRASP no algoritmo proposto sem dois componentes básicos do MPNS-GRASP2, os procedimentos RBLK e PR, era óbvio que um algoritmo computacionalmente mais eficiente seria aplicado , mas provavelmente um algoritmo que não seria capaz de dar resultados tão bons e competitivos quanto os do MPNS-GRASP2. Porém, não apenas foram encontrados resultados competitivos, mas em todas as instâncias foram encontrados melhores resultados no HBMOTSP, do MPNS-GRASP2. Assim, o uso da Otimização de Acasalamento das Abelhas Melíferas

Tabela 3
Classificação dos algoritmos com base na qualidade média (%).

Método	Média	Método	Média
Tour mesclando [5]	0,0034	3opt-J [37]	3.392
PHGA [61]	0,0066	TS-2opt-DB [85]	3,698
HBMO para TSP	0,0102	4-Hyperopt [37]	3,783
ILK-NYYY [38]	0,0354	GÊNIO [24]	3,98
MTV de KH na LK [35]	0,0367	3-Hyperopt [37]	4.185
ILK-J [37]	0,0853	TS-2opt-2opt [85]	4.763
ALK [38]	0,48	2.5opt-B [10]	4,8585
Concorde CLK [5]	0,4863	2opt-J [37]	5,345
I-3-Opt-J [37]	0,514	2-Hyperopt [37]	5,386
ACR Acorrentado LK [5]	0,5827	2opt-B [10]	6.091
ILK-N [60]	0,5924	HK-Christo fi des [38]	6,523
Helsgaun LK [35]	0,669	GENI [24]	7,88
BSDPH [38]	0,710	CCA [38]	10,291
LK-HK-Christo-começa [38]	1.051	CW [13]	10.594
TS-LK-DB [85]	1.060	FI [10]	14.623
TS-SC-DB [85]	1,111	2opt-C [5]	14,946
ENS-GRASP [54]	1.181	RI [10]	15,48
LK-NYYY [38]	1.228	C-Greedy [5]	16,741
Johnson LK [37]	1.388	Boruvka [5]	17.063
TS-LK-LK [85]	1,475	B-Greedy [38]	17,237
TS-SC-SC [85]	1.483	CHCI [10]	17,4185
Neto LK [60]	1.547	Q-Boruvka [5]	18,2487
SC EC [38]	1.660	NN [5]	24.916
VNS-3-Hyperopt [38]	2.061	Melhor - Way Strip [38]	48,0885
Concorde-LK [5]	2.778	Faixa [38]	63,52
VNS-2-Hyperopt [38]	2.874	FRP [10]	64,3185
Applegate LK [5]	3,20	Preenchimento de espaço [38]	70,106
3opt-B [10]	3.288		

algoritmo em um algoritmo de hibridização dá resultados muito bons. Com relação ao HybGEN e ao HBMOTSP, deve-se notar que, embora ambos sejam algoritmos baseados em população e usem o mesmo operador de cruzamento e ENS-GRASP, o HBMOTSP dá melhores resultados do que o HybGEN.

Os resultados do algoritmo HBMOTSP proposto também são comparados com aqueles apresentados no Desafio de Implementação DIMACS (http://www.resea dsj / chtsp /) Este desafio é provavelmente o exame mais extenso até hoje de algoritmos heurísticos no campo do TSP e apresenta resultados computacionais para aproximadamente 40 heurísticas de construção de tour e para aproximadamente 80 heurísticas de melhoria de tour. No desafio, as heurísticas de melhoria do tour são classificadas em quatro categorias com base em algumas características específicas. A primeira categoria consiste nos algoritmos clássicos de busca local, como 2opt e 3-opt, enquanto a segunda consiste em algoritmos que são implementações diferentes do algoritmo de Lin-Kernighan. Na terceira categoria pertencem os algoritmos que usam Iterado ou Chained Lin – Kernighan, enquanto na última categoria estão incluídos os algoritmos de metaheurística, como Busca Tabu e Busca de Vizinhança Variável. Dentro Tabela 3 , a classificação de todos os algoritmos com base em sua qualidade média e para as instâncias com número de nó maior que 1000 é apresentada. O algoritmo HBMOTSP proposto está classificado em terceiro lugar entre 55 algoritmos. O algoritmo é classificado melhor do que todas as heurísticas de construção de tour. É, também, classificado melhor do que todos os algoritmos de pesquisa local simples. É melhor classificada do que todas as implementações de pesquisa de vizinhança variável e, também, é melhor classificada do que todas as metaheurísticas. Para obter mais detalhes sobre a classificação dos algoritmos, consulte [54] . Deve-se notar que uma comparação justa em termos de eficiência computacional é difícil porque a velocidade computacional é afetada, principalmente, pelo compilador e o hardware que são usados, portanto, as comparações com os algoritmos da literatura são realizadas apenas em termos de qualidade das soluções.

5. Conclusões e pesquisas futuras

Neste artigo, uma abordagem inspirada na natureza foi introduzida para o tratamento eficaz do Problema do Caixeiro Viajante Euclidiano (TSP). Mais especificamente, uma metodologia híbrida inspirada na natureza algorítmica foi proposta, ou seja, o algoritmo Honey Bees Mating Optimization para o TSP (HBMOTSP). Uma das principais contribuições deste trabalho foi mostrar que a Otimização de Acasalamento das Abelhas Melíferas pode ser utilizada em síntese híbrida com outras metaheurísticas para a solução do Problema do Caixeiro Viajante com resultados notáveis tanto de qualidade quanto de eficiência computacional. O algoritmo foi aplicado em um conjunto de instâncias de benchmark do TSPLIB e deu resultados muito satisfatórios. Nossas pesquisas futuras serão focadas principalmente na aplicação do algoritmo de otimização de acasalamento das abelhas melíferas em outros problemas de otimização combinatória.

Referências

[1] HA Abbass, Amonogenous MBO approach to satisfability, in: Proceeding of the International Conference on Computational Intelligence for Modeling, Controle e Automação, CIMCA'2001, Las Vegas, NV, EUA, 2001.
[2] HA Abbass, Marriage in honey-bee optimization (MBO): a haplometris polygynous swarming approach, em: The Congress on Evolutionary Computation (CEC2001), Seoul, Korea, maio de 2001, 2001, pp. 207–214.
[3] A. Afshar, O. Bozog Haddad, MA Marino, BJ Adams, algoritmo de otimização de acasalamento de abelha melífera (HBMO) para operação de reservatório ideal, Journal of the Franklin Institute 344 (2007) 452–462.
[4] N. Ansari, E. Hou, Computational Intelligence for Optimization, primeira edição, Kluwer Academic Publishers, Boston, 1997.
[5] D. Applegate, R. Bixby, V. Chvatal, W. Cook, Chained Lin – Kernighan para grandes problemas de caixeiro viajante, Informs Journal on Computing 15 (2003) 82–92
[6] A. Badr, A. Fahmy, Uma prova de convergência para algoritmos de formigas, Information Sciences 160 (2004) 267-279.
[7] Y. Bai, W. Zhang, Z. Jin, Uma nova estratégia de mapas auto-organizáveis para resolver o problema do caixeiro viajante, Chaos, Solitons and Fractals 28 (4) (2006) 1082–1089.
[8] R. Baralia, JI Hildago, R. Perego, Uma heurística híbrida para o problema do caixeiro viajante, IEEE Transactions on Evolutionary Computation 5 (6) (2001) 1–41
[9] A. Baykasoglu, L. Ozbakor, P. Tapkan, Arti fi cial bee colony algoritmo e sua aplicação ao problema de atribuição generalizada, em: FTS Chan, MK Tiwari (Eds.), Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, I-Tech Education and Publishing, 2007, pp. 113-144. [10] JL Bentley, Fast algorítmos para problemas geométricos do caixeiro-viajante, ORSA Journal on Computing 4 (1992) 387–411. [11] Y. Chen, P. Zhang, recozimento otimizado do problema do caixeiro viajante do n distribuição do vizinho mais próximo, Física A: Estatística e Theoretical Physics 371 (2) (2006) 627–632.
[12] SC Chu, JF Roddick, JS Pan, sistema de colônia de formigas com estratégias de comunicação, Ciências da Informação 167 (1-4) (2004) 63-76.
[13] G. Clarke, JW Wright, Programação de veículos de um depósito central para uma série de pontos de entrega, Operations Research 12 (1964) 568–581. [14] D. Dasgupta (Ed.), Arti fi cial Immune Systems and their Application, Springer, Heidelberg, 1998.
[15] LD De Castro, J. Timmis, Arti fi cial Immune Systems: A New Computational Intelligence Approach, Springer, Heidelberg, 2002.
[16] M. Dorigo, LM Gambardella, Sistema de colônia de formigas: uma abordagem de aprendizagem cooperativa para o problema do caixeiro viajante, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[17] M. Dorigo, LM Gambardella, colônias de formigas para o problema do caixeiro viajante, Biosystems 43 (1997) 73–81.
[18] M. Dorigo, T. Stutzle, Ant Colony Optimization, A Bradford Book, The MIT Press Cambridge, Massachusetts, Londres, Inglaterra, 2004.
[19] H. Drias, S. Sadeg, S. Yahi, Cooperative abelhas enxame para resolver o problema de satisfação com peso máximo, em: IWAAN International Work Conference on Arti fi cial and Natural Neural Networks, LNCS, vol. 3512, 2005, pp. 318–325.
[20] I. Ellabib, P. Calamai, O. Basir, estratégias de troca para sistema de colônia de formigas múltiplas, Ciências da Informação 177 (2007) 1248-1264.
[21] M. Fathian, B. Amiri, A. Maroosi, Aplicação do algoritmo de otimização de acasalamento de abelhas em agrupamento, Matemática Aplicada e Computação 190 (2007) 1502–1513.
[22] TA Feo, MGC Resende, Greedy randomized randomized adaptive search procedure, Journal of Global Optimization 6 (1995) 109–133. [23] R. Garfinkel, G. Nemhauser, Integer Programming, Wiley and Sons, 1972.
[24] M. Gendreau, A. Hertz, G. Laporte, Novos procedimentos de inserção e pós-otimização para o problema do caixeiro viajante, Operations Research 40 (1992) 1086–1094.

[25] F. Glover, Future path for integer programming and links to arti fi cial intelligence, Computers and Operations Research 13 (1986) 533-549. [26] F. Glover, Tabu search I, ORSA Journal on Computing 1 (3) (1989) 190-206.
[27] F. Glover, Tabu search II, ORSA Journal on Computing 2 (1) (1990) 4-32.
[28] Glover, F., 1990. Tabu search: A tutorial, Center for Applied Arti fi cial Intelligence, University of Colorado, 1-47. [29] F. Glover, G. Konchenberger, Handbook of Metaheuristics, Kluwer Academic Publishers, Dordrecht, 2003.
[30] F. Glover, M. Laguna, Tabu search, em: PM Pardalos, MGC Resende (Eds.), Handbook of Applied Optimization, Oxford University Press, 2002, pp. 194–209.
[31] EFG Goldbarg, GR Souza, MC Goldbarg, Otimização do enxame de partículas para o problema do caixeiro viajante, em: EVOCOP 2006, LNCS, vol. 3906, 2006, pp. 99-110.
[32] G. Gutin, A. Punnen, O Problema do Caixeiro Viajante e suas Variações, Kluwer Academic Publishers, Dordrecht, 2002.
[33] OB Haddad, A. Afshar, MA Marino, algoritmo de otimização de acasalamento de abelhas (HBMO): uma nova abordagem heurística para otimização de recursos hídricos, Water Resources Management 20 (2006) 661–680.
[34] P. Hansen, N. Mladenovic, Pesquisa de vizinhança variável: princípios e aplicações, European Journal of Operational Research 130 (2001) 449-467. [35] K. Helsgaum, Uma implementação eficaz da heurística do caixeiro viajante Lin-Kernighan, European Journal of Operational Research 126 (2000) 106-130.
[36] DS Johnson, CH Papadimitriou, Complexidade computacional, em: EL Lawer, JK Lenstra, AHD Rinnoy Kan, DB Shmoys (Eds.), The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley and Sons, 1985, pp. 37-85.
[37] DS Johnson, LA McGeoch, O problema do caixeiro viajante: um estudo de caso, em: E. Aarts, JK Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley and Sons, 1997, pp. 215–310.
[38] DS Johnson, LA McGeoch, Análise experimental de heurísticas para o STSP, em: G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht, 2002, pp. 369–444.
[39] DS Johnson, G. Gutin, LA McGeoch, A. Yeo, W. Zhang, A. Zverovitch, Análise experimental de heurísticas para o ATSP, em: G. Gutin, A. Punnen (Eds.), O problema do caixeiro viajante e suas variações, Kluwer Academic Publishers, Dordrecht, 2002, pp. 445-487.
[40] D. Karaboga, B. Basturk, Um algoritmo poderoso e eficiente para otimização de função numérica: algoritmo arti fi cial bee colony (ABC), Journal of Global Optimization 39 (2007) 459–471.
[41] D. Karaboga, B. Basturk, No desempenho da colônia de abelhas arti fi cial (ABC) algoritmo, Applied Soft Computing 8 (2008) 687-697.
[42] J. Kennedy, R. Eberhart, Particle swarm optimization, em: Proceedings of 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942-1948.
[43] S. Kirkpatrick, CD Gelatt, MP Vecchi, Optimization by simulated annealing, Science 220 (1982) 671-680.
[44] G. Laporte, O problema do caixeiro viajante: uma visão geral de algoritmos exatos e aproximados, European Journal of Operational Research 59 (1992) 231–247.
[45] EL Lawer, JK Lenstra, AHG Rinnoy Kan, DB Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley e Sons, 1985.
[46] FX Le Louarn, M. Gendreau, JY Potvin, GENI formigas para o problema do caixeiro viajante, Annals of Operations Research 131 (2004) 187–201.
[47] X. Li, P. Tian, J. Hua, N. Zhong, Otimização de enxame de partículas discretas híbridas para o problema do caixeiro viajante, em: SEAL 2006, LNCS, vol. 4247, 2006, pp. 181–188.
[48] S. Lin, Soluções de computador do problema do caixeiro viajante, Bell Systems Technical Journal 44 (1965) 2245–2269.
[49] S. Lin, BW Kernighan, Um algoritmo heurístico eficaz para o problema do caixeiro viajante, Operation Research 21 (1973) 498-516.
[50] A. Liu, Z. Deng, S. Shan, Sistema de formigas de contribuição média: uma versão melhorada de otimização de colônia de formigas para o problema do caixeiro viajante, em: SEAL 2006, LNCS, vol. 4247, 2006, pp. 489–496.
[51] SJ Louis, G. Li, Case injetou algoritmos genéticos para problemas do caixeiro-viajante, Information Sciences 122 (2000) 201–225.
[52] T. Machado, H. Lopez, Um modelo de otimização de enxame de partículas híbrido para o problema do caixeiro viajante, Adaptive and Natural Computing Algorithm (2005) 255–258.
[53] M. Manfrin, M. Birattari, T. Stutzle, M. Dorigo, Otimização de colônia de formigas paralelas para o problema do caixeiro viajante, em: ANTS 2006, LNCS, vol. 4150, 2006, pp. 224–234.
[54] Y. Marinakis, A. Migdalas, PM Pardalos, Expansão da vizinhança GRASP para o problema do caixeiro viajante, Otimização computacional e Applications 32 (2005) 231–257.
[55] Y. Marinakis, A. Migdalas, PM Pardalos, Um algoritmo híbrido Genetic-GRASP usando relaxamento Lagrangeano para o problema do caixeiro viajante, Journal of Combinatorial Optimization 10 (2005) 311–326.
[56] Y. Marinakis, A. Migdalas, PM Pardalos, Pesquisa de vizinhança de fase múltipla GRASP baseada em relaxamento Lagrangeano e retrocesso aleatório Lin-Kernighan para o problema do caixeiro viajante, Journal of Combinatorial Optimization 17 (2009) 134–156.
[57] TAS Masutti, LN de Castro, Uma rede neural auto-organizada usando ideias do sistema imunológico para resolver o problema do caixeiro viajante, Ciências da Informação 179 (2009) 1454–1468.
[58] N. Mladenovic, P. Hansen, pesquisa de vizinhança variável, Computers and Operations Research 24 (1997) 1097-1100.
[59] A. Modares, S. Somhom, T. Enkawa, Uma abordagem de rede neural auto-organizada para vários caixeiros-viajantes e problemas de roteamento de veículos, International Transactions in Operational Research 6 (6) (1999) 591–606.
[60] DM Neto, Compensação eficiente de cluster para heurísticas de Lin-Kernighan, Ph.D. Tese, Universidade de Ciência da Computação de Toronto, Canadá, 1999.
[61] HD Nguyen, I. Yoshihara, K. Yamamori, M. Yasunaga, Implementação de um GA híbrido eficaz para problemas de caixeiros-viajantes em grande escala, IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetic 37 (2007) 92–99.
[62] M. Ninio, JJ Schneider, Weight annealing, Physica A: Statistical and Theoretical Physics 349 (3-4) (2005) 649-666.
[63] DT Pham, E. Kog, A. Ghanbarzadeh, S. Otri, S. Rahim, M. Zaidi, O algoritmo das abelhas - Uma nova ferramenta para problemas de otimização complexos, em: IPROMS 2006 Proceedings of Second International Virtual Conference on Intelligent Production Machines and Systems, Oxford, Elsevier, 2006.
[64] JY Potvin, Algoritmos genéticos para o problema do caixeiro viajante. Metaheurísticas em otimização combinatória, Annals of Operations Research 63 (1996) 339–370.
[65] L. Qu, R. Sun, Uma abordagem sinérgica para algoritmos genéticos para resolver o problema do caixeiro viajante, Information Sciences 117 (1999) 267-283.
[66] C. Rego, F. Glover, Local search and metaheuristics, em: G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht, 2002, pp. 309-367.
[67] G. Reinelt, The Traveling Salesman, Computational Solutions for TSP Applications, Springer-Verlag, Berlin, 1994.
[68] JF Rennard, Manual de pesquisa em computação inspirada na natureza para economia e gestão, Idea Group Reference, Hershey-PA, EUA, 2006. [69] MGC Resende, CC Ribeiro, Greedy randomized randomized adaptive search procedures, in: F. Glover , GA Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, 2003, pp. 219–249.
[70] Y. Rochat, ED Taillard, Probabilistic diversi? Cation and intensi? Cation in local search for Vehicle Routing, Journal of Heuristics 1 (1995) 147-167. [71] S. Ronald, Routing and scheduling problems, em: L. Chambers (Ed.), Practical handbook of genetic algoritms - Applications, CRC Press, New York, 1995, pp. 367–430.
[72] DJ Rosenkratz, RE Stearns, PM Lewis, Uma análise de várias heurísticas para o problema do caixeiro viajante, SIAM Journal on Computing 6 (1977) 563–581.
[73] XH Shi, YC Liang, HP Lee, C. Lu, QX Wang, algoritmos baseados em otimização de enxame de partículas para TSP e TSP generalizado, processamento de informações Letters 103 (2007) 169–176.

[74] PH Siqueira, M. Teresinha, A. Steiner, S. Scheer, Uma nova abordagem para resolver o problema do caixeiro viajante, Neurocomputing 70 (4–6) (2007) 1013–1021.
[75] ED Taillard, Ant systems, em: PM Pardalos, MGC Resende (Eds.), Handbook of Applied Optimization, Oxford University Press, 2002, pp. 130–138. [76] J. Teo, HA Abbass, Uma abordagem de recozimento verdadeiro para o casamento em algoritmo de otimização de abelhas, International Journal of Computational Intelligence and Applications 3 (2) (2003) 199–211.
[77] D. Teodorovic, M. Dell'Orco, Otimização de colônias de abelhas - Uma abordagem de aprendizagem cooperativa para problemas de transporte complexos, OR avançado e IA Methods in Transportation (2005) 51–60.
[78] CK Ting, ST Li, C. Lee, Sobre a estratégia de acasalamento harmonioso por meio da pesquisa tabu, Information Sciences 156 (2003) 189–214.
[79] CF Tsai, CW Tsai, CC Tseng, Uma nova abordagem heurística híbrida para resolver grandes problemas de caixeiros-viajantes, Ciências da Informação 166 (2004) 6781. [80] HK Tsai, JM Yang, YF Tsai, CY Kao, Um algoritmo evolucionário para grandes problemas de caixeiro viajante, IEEE Transactions on Systems, Man, e Cybernetics - Parte B: Cybernetics 34 (2004) 1718–1729.
[81] A. Ugur, Path planning on a cuboid using genetic algorítmos, Information Sciences 178 (2008) 3275-3287.
[82] Y. Wang, XY Feng, YX Huang, DB Pu, WG Zhou, YC Liang, CG Zhou, Um novo algoritmo evolutivo de enxame quântico e suas aplicações, Neurocomputing 70 (4–6) (2007) 633–640.
[83] HF Wedde, M. Farooq, Y. Zhang, BeeHive: um algoritmo de roteamento tolerante a falhas e fi ciente inspirado no comportamento das abelhas melíferas, em: M. Dorigo (Ed.), Ant Colony Optimization and Swarm Intelligence, LNCS, vol. 3172, Springer, Berlin, 2004, pp. 83-94.
[84] XS Yang, Otimizações de engenharia via algoritmos de abelhas virtuais inspirados na natureza, em: JM Yang, JR Alvarez (Eds.), IWINAC, LNCS, vol. 3562, Springer-Verlag, Berlin Heidelberg, 2005, pp. 317-323.
[85] M. Zachariasen, M. Dam, Tabu pesquisa sobre o problema geométrico do caixeiro viajante, em: IH Osman, JP Kelly (Eds.), Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, Boston, 1996, pp. 571–587.