FUNDAMENTOS DE OTIMIZAÇÃO POR INTELIGÊNCIA DE ENXAMES: UMA VISÃO GERAL

Adriane Beatriz de Souza Serapião*
adriane@rc.unesp.br

*UNESP /IGCE/DEMAC
Caixa Postal 178
CEP 13506-900 - Rio Claro (SP)

RESUMO

Este artigo apresenta uma breve revisão de alguns dos mais recentes métodos bioinspirados baseados no comportamento de populações para o desenvolvimento de técnicas de solução de problemas. As metaheurísticas tratadas aqui correspondem às estratégias de otimização por colônia de formigas, otimização por enxame de partículas, algoritmo *shuffled frog-leaping*, coleta de alimentos por bactérias e colônia de abelhas. Os princípios biológicos que motivaram o desenvolvimento de cada uma dessas estratégias, assim como seus respectivos algoritmos computacionais, são introduzidos. Duas aplicações diferentes foram conduzidas para exemplificar o desempenho de tais algoritmos. A finalidade é enfatizar perspectivas de aplicação destas abordagens em diferentes problemas da área de engenharia.

PALAVRAS-CHAVE: inteligência computacional, computação evolutiva, computação natural, computação bioinspirada, inteligência coletiva, algoritmos de otimização.

ABSTRACT

This paper presents an overview of some most recent bio-inspired methods based on swarm behaviors for the development of problem-solving techniques. The metaheuristics provided here are ant colony optimization, particle swarm optimization, shuffled frog-leaping algorithm, bacterial foraging optimization and bee colony. The basic biological principles that have motivated the development of each strategy, as well as their computational algorithms, are introduced. Two different applications were carried out in order to clarify the performance of such algorithms. The goal is to emphasize perspectives of applications of these approaches in different engineering problems.

KEYWORDS: computational intelligence, evolutionary computing, natural computing, bio-inspired computing, swarm intelligence; optimization algorithms.

1 INTRODUÇÃO

Não-linearidades e interações complexas entre variáveis de projeto (x) e variáveis operacionais ($g(x) \leq 0$, $h(x) = 0$) em problemas de engenharia (f) formam um espaço de busca (S) que pode conter várias soluções ótimas ($x^* \mid f(x^*) < f(x), \forall \; x \in S \subset \Re^n$), como no caso de um problema de minimização. Por causa da possibilidade de encontrar mínimos locais ou soluções subótimas, os métodos baseados em gradiente podem não ser bons candidatos como algoritmos de otimização eficientes quando aplicados a uma ampla gama de projetos de engenharia e de problemas operacionais.

Por outro lado, muitos problemas de engenharia também não podem ser tratados através de métodos analíticos, seja por causa da dificuldade de formulação da modelagem ou do esforço matemático exigido na solução, principalmente quando estão envolvidas funções não-diferenciáveis ou descontínuas.

Neste sentido, nos últimos tempos, algoritmos bioinspirados baseados em populações e metaheurísticas vêm sendo usa-

Artigo submetido em 23/10/2008 (Id.: 00909)
Revisado em 06/02/2009, 28/03/2009, 14/04/2009, 20/04/2009
Aceito sob recomendação do Editor Associado Prof. Ivan Nunes Da Silva

dos para resolver problemas de busca e otimização em vários domínios de problemas para os quais soluções robustas são difíceis ou impossíveis de encontrar usando abordagens tradicionais como a programação matemática. O princípio fundamental desses algoritmos utiliza um método construtivo para a obtenção da população inicial (soluções factíveis iniciais) e uma técnica de busca local para melhorar a solução da população, considerando que os indivíduos (soluções) dessa população são evoluídos de acordo com regras especificadas que consideram o intercâmbio de informações entre os indivíduos. Esse processo conduz a população em direção à obtenção de uma solução ótima. Tais algoritmos são conhecidos como algoritmos de computação evolutiva.

Duas abordagens evolutivas baseadas em populações têm se destacado: algoritmos evolutivos e algoritmos de enxames. Os Algoritmos Evolutivos (AE) tradicionalmente incluem Algoritmos Genéticos (AG) (Goldberg, 1989), Programação Evolutiva (PE), Estratégias Evolutivas (EE) (De Jong, 2006) e Programação Genética (PG) (Koza, 2003). Algoritmos mais recentemente desenvolvidos como Algoritmos de Estimação de Distribuição (AED) (Pelikan, 2006) e Algoritmos Genéticos Competentes (Goldberg, 2002) também são AE.

Os AEs têm sido intensamente estudados e amplamente aplicados para resolver vários problemas científicos e de engenharia, tais como projeto de robô, despacho econômico de sistemas de energia e problemas de identificação de dobra de proteína, só para mencionar alguns. Estes algoritmos têm desfrutado de sucesso nas aplicações devido à sua simplicidade, robustez e flexibilidade. Os AEs atuam sobre uma população de possíveis soluções aplicando o princípio de diversidade de indivíduos e da sobrevivência de indivíduos mais fortes e bem adaptados ao ambiente, que se reproduzem através de operadores que imitam os conceitos genéticos, criando descendentes mais fortes que se aproximam da solução do problema.

Este artigo explora as abordagens evolutivas baseadas em algoritmos de enxames, conforme apresentado na seqüência. A Inteligência de Enxames, também referenciada como Inteligência de Colônias ou Inteligência Coletiva, é um conjunto de técnicas baseadas no comportamento coletivo de sistemas auto-organizados, distribuídos, autônomos, flexíveis e dinâmicos. Estes sistemas são formados por uma população de agentes computacionais simples que possuem a capacidade de perceber e modificar o seu ambiente de maneira local. Esta capacidade torna possível a comunicação entre os agentes, que captam as mudanças no ambiente geradas pelo comportamento de seus congêneres. Embora não exista uma estrutura centralizada de controle que estabelece como os agentes devem se comportar, e mesmo não havendo um modelo explícito do ambiente, as interações locais entre os agentes geralmente levam ao surgimento de um comportamento global que se aproxima da solução do problema.

As propriedades principais de um sistema de inteligência de enxame são (Millonas, 1994):

- *Proximidade* – os agentes devem ser capazes de interagir;
- *Qualidade* – os agentes devem ser capazes de avaliar seus comportamentos;
- *Diversidade* – permite ao sistema reagir a situações inesperadas;
- *Estabilidade* – nem todas as variações ambientais devem afetar o comportamento de um agente;
- *Adaptabilidade* – capacidade de adequação a variações ambientais.

Duas principais linhas de pesquisa que emergem dessas propriedades podem ser observadas na inteligência de enxames: trabalhos inspirados no estudo do comportamento de insetos sociais, como formigas, abelhas, cupins e vespas; e trabalhos inspirados na habilidade das sociedades humanas em processar conhecimento.

As técnicas mais conhecidos de Inteligência de Enxames são a otimização por colônia de formigas, otimização por enxame de partículas, algoritmo *shuffled frog-leaping*, algoritmos de coleta de alimentos por bactérias e algoritmos de colônia de abelhas, que por sua vez serão foco deste artigo.

A Inteligência de Enxames também é considerada um ramo da abordagem computacional conhecida como *Computação Natural*. Tal abordagem é a versão computacional do processo de extrair ideias da natureza para desenvolver sistemas computacionais. Uma visão abrangente de conceitos, algoritmos e aplicações da computação natural pode ser encontrada em (de Castro, 2006).

Neste artigo apresenta-se uma revisão dos principais conceitos relativos aos algoritmos mais importantes das técnicas de Inteligência de Enxames acima mencionados, juntamente com algumas de suas variantes e aplicações. O objetivo é motivar o uso desses algoritmos no desenvolvimento efetivo de trabalhos em diferentes áreas da engenharia e de automação de processos. Dois experimentos distintos – otimização de funções e despacho econômico de carga – foram conduzidos para indicar aplicações possíveis das metaheurísticas em problemas da engenharia e para caracterizar as principais diferenças encontradas nos métodos.

2 OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS

O comportamento de coleta de alimentos de muitas sociedades de formigas (*L. humilis, Linepithema humile, Lasius niger*) baseia-se na comunicação indireta mediada por feromônios. Esse tipo de comunicação através de marcas ou signos é chamado *estigmergia*. Enquanto caminham do formigueiro para as fontes de alimento e vice-versa, as formigas depositam feromônios no chão, formando uma trilha química. As formigas são capazes de liberar a substância e tendem a escolher caminhos onde haja maior concentração química. A trilha química é uma estrutura emergente e auto-organizada e resulta de um *feedback* positivo. Quanto mais química uma trilha, mais formigas são atraídas, o que a torna ainda mais química, reforçando o caminho que novamente atrai ainda mais formigas, em uma espiral crescente.

Deneubourg e colaboradores (1990) realizaram um experimento com formigas reais *L. humile*, observando o comportamento delas na coleta de alimentos. Foi colocado um ninho de formigas em um aquário com uma fonte de alimentos na outra ponta. Para chegar até esse alimento foram criados dois caminhos, sendo um maior que o outro. Inicialmente cada formiga segue um caminho exploratório aleatório. Como as formigas que escolheram o menor caminho faziam o percurso mais rapidamente que as outras, elas depositavam uma maior quantidade de feromônio nesse caminho em relação ao outro em um mesmo intervalo de tempo. Logo, em um determinado momento a intensidade do feromônio no caminho mais curto estava tão alta que quase todas as formigas seguiam por ele.

Inspirados pelos experimentos de coleta de alimentos por formigas, Dorigo *et al.* (1991) desenvolveram um modelo computacional com tais conceitos para resolver o problema do caixeiro viajante. O algoritmo proposto baseia-se em um grupo de "formigas artificiais" que liberam feromônio durante o seu trajeto e seguem "trilhas de feromônio artificial" para encontrar o menor caminho entre todas as cidades.

Para solucionar o problema, há uma colônia de formigas, onde cada formiga vai de uma cidade a outra independentemente, buscando cidades próximas ou caminhando aleatoriamente. Durante o percurso, uma formiga segue um caminho qualquer entre cidades, determinado por uma regra probabilística que sugere o caminho mais promissor, e libera uma determinada quantidade de feromônio. Essa quantidade é inversamente proporcional ao comprimento total do caminho percorrido pela formiga. Assim que todas as formigas tiverem concluído suas rotas e liberado feromônio, as conexões entre cidades que fazem parte da maior quantidade de rotas mais curtas terão uma quantidade maior de feromônio depositado. Visto que o feromônio evapora com o decorrer do tempo, quanto maior for o comprimento do caminho, mais rápido será o desaparecimento de uma trilha em um caminho longo.

Os algoritmos de otimização baseados em colônias de formigas foram inicialmente utilizados para resolver problemas típicos de otimização combinatória que podem ser representados em grafos (Dorigo *et al*.,1991; Dorigo *et al*., 1996).

2.1 Algoritmo ACO

O termo algoritmo de otimização por colônia de formigas (*Ant Colony Optimization* – ACO) é um termo genérico que designa um procedimento geral de uma classe de metaheurísticas baseadas no comportamento de formigas. Assim, embora todo algorimo ACO seja um algoritmo de formiga, o contrário não é verdadeiro.

Ant System (AS) foi o primeiro algoritmo baseado no comportamento de formigas desenvolvido por Dorigo *et al.* (1991). Inicialmente três diferentes variantes foram propostas: *AS-density*, *AS-quantity* e *AS-cycle*, diferindo-se pela maneira na qual as trilhas de feromônio eram atualizadas (Dorigo *et al*., 1996). A última variante mostrou-se mais eficiente e a maior parte dos algoritmos de otimização por colônia de formigas atuais derivam-se dela.

O modelo matemático do comportamento de uma colônia de formigas em busca de alimentos é descrito a seguir. Dado um grafo G com V vértices representando um problema, uma formiga artificial é colocada em cada um dos vértices. Cada formiga percorre um caminho seguindo uma fórmula probabilística em função do feromônio "depositado" em cada aresta do grafo para chegar ao destino.

Soluções parciais do problema são chamadas de estado. Cada formiga muda de um estado anterior para o próximo estado, que corresponde a uma solução parcial mais completa, com o objetivo de chegar ao estado final que é a solução total do problema. Em cada passo da construção da solução a formiga visualiza o conjunto de expansões possíveis para a solução atual, isto é, identifica o conjunto de estados viáveis para o qual se pode passar a partir do estado atual.

Cada formiga possui uma estrutura de dados chamada *lista tabu*, que guarda os vértices já visitados e proíbe que a formiga o visite novamente até completar o caminho (encontrar a fonte de alimento). Após a construção de todos os caminhos a intensidade de feromônio em cada aresta (trilha) é acrescida de forma proporcional à qualidade da solução gerada.

Para construir a solução cada formiga (k) utiliza iterativamente uma regra de transição de estado ($p_{i,j}$) (função probabilística) para decidir se incluirá ou não determinada aresta

na solução:

$$p_{i,j}^{k}(t) = \begin{cases} \frac{\tau_{i,j}^{\alpha}(t) \times \eta_{i,j}^{\beta}}{\sum_{j \in J^k} \tau_{i,j}^{\alpha}(t) \times \eta_{i,j}^{\beta}}, & \text{se } j \in J^k \\ 0, & \text{caso contrário,} \end{cases} \quad (1)$$

onde:τ_{ij} é a trilha de feromônio da combinação (i, j), η_{ij} é a heurística local da combinação (i, j) (visibilidade), α é a importância relativa da trilha de feromônio e β é a importância relativa da heurística local.

Para atualizar a trilha de feromônio nas arestas é calculada inicialmente a quantidade a ser depositada em cada uma delas proporcionalmente à qualidade das soluções que pertencem a elas.

$$\tau_{i,j}(t+1) = (1 - -\rho) \times \tau_{i,j}(t) + \rho \times \Delta\tau_{ij}, \quad (2)$$

$$\Delta\tau_{i,j} = \begin{cases} \frac{1}{f(S)}, & \text{se } (i,j) \in S \\ 0, & \text{caso contrário,} \end{cases} \quad (3)$$

onde: ρ é a persistência da trilha (taxa de evaporação), usada para evitar a rápida convergência das formigas em uma região do espaço de busca, e $\Delta\tau_{i,j}$ é a quantidade de feromônio que será depositada na aresta (i, j), que depende da função de custo f da solução S. A equação do cálculo da quantidade de feromônio $(\Delta\tau_{i,j})$ acima representa a heurística adotada para o problema do caixeiro viajante (o inverso do comprimento total do caminho S), porém ela pode ser modificada para corresponder aos requisitos do problema tratado.

O valor de τ é modificado em cada iteração do processo de busca para aumentar os valores dos movimentos que resultaram em uma boa solução (correspondendo ao depósito de feromônio no caminho) e diminuir todos os demais valores (correspondente à evaporação do feromônio).

Gutjahr (2000, 2002) provou que um sistema ACO genérico converge para a solução ótima em certas condições. Stutzle & Dorigo (2002) também mostraram um outro tipo de prova de convergência do algoritmo ACO.

O algoritmo de otimização por colônia de formigas pode ser resumido pelo procedimento geral, em sua forma mais simples, descrito no Algoritmo 1.

Algoritmo 1: Procedimento geral de um ACO.

1. Inicialize os parâmetros k, ρ, α, β e as trilhas de feromônio τ_{ij} com o mesmo valor inicial τ_0.
2. Coloque cada formiga k em uma aresta aleatoriamente selecionada do grafo $G(V,E)$, onde V é o conjunto de vértices e E é o conjunto de arestas de G.
3. Para cada formiga k, em cada aresta até o destino, construa soluções baseadas na *regra de transição de estado* $(p_{i,j}^{k})$.
4. Avalie o custo de todas as soluções (calcule $f(S)$) .
5. Guarde a melhor solução até o momento.
6. Para cada aresta do grafo, aplique a regra de atualização da trilha de feromônio (calcule $\Delta\tau_{i,j}$ e $\tau_{i,j}$).
7. Se condição de término não for alcançada, retorne ao passo 3.

2.2 Avanços e aplicações do algoritmo ACO

O algoritmo ACO original para solução de problemas combinatoriais é conhecido como *Ant System* (AS) (Dorigo *et al*., 1991). Desde então vários outros algoritmos ACO foram introduzidos compartilhando dos mesmos princípios. A seguir, algumas das variações mais conhecidas são comentadas.

O algoritmo ANT-Q (Gambardella & Dorigo, 1995) é uma extensão do AS que integra algumas ideias de *Q-learning* (Watkins, 1989). No ANT-Q a atualização da trilha de feromônio usando-se um valor que é uma previsão do valor do próximo estado. Essa abordagem acabou sendo abandonada posteriormente, pois o algoritmo ACS, mais simples, conseguia um desempenho equivalente. O *Ant Colony System* (ACS) introduziu a atualização do feromônio local (atualização *online*) além da atualização do feromônio realizada no fim do processo de construção do caminho (atualização *offline*) (Gambardella & Dorigo, 1996). A atualização *online* é realizada em cada passo de construção para a última aresta percorrida.

No *AS Elitista* (Dorigo *et al*., 1996) apenas a melhor solução gerada até o momento tem a trilha de feromônio atualizada. O algoritmo *MAX-MIN AS* (MMAS) (Stutzle & Hoos, 1996) considera que apenas a melhor formiga a cada passo atualiza as trilhas de feromônio e que a quantidade do feromônio é limitada por um valor mínimo e um valor máximo.

O *Rank-Based AS* (Bullheimer *et al*., 1997) atualiza as trilhas de feromônio de acordo com uma função de classifi-

cação ponderada do desempenho de cada formiga na busca da solução. Formigas com os melhores desempenhos têm a maior taxa de atualização da trilha de feromônio. O algoritmo *Approximate Nondeterministc Tree-Search* (ANTS) (Maniezzo, 1999) modificou o modo de cálculo da atualização da trilha de feromônio e combinou ACO com técnicas de limites inferiores de programação matemática para eliminar extensões de soluções parciais que levariam a custos maiores que a melhor solução já encontrada (atratividade). Este mecanismo torna o algoritmo similar em estrutura aos algoritmos de busca em árvore.

O *Best Worst Ant System* (BWAS) (Cordón *et al.*, 2000) incorpora conceitos de computação evolutiva (Goldberg, 1989). O algoritmo reforça as trilhas da melhor solução global e penaliza cada trilha da pior solução gerada na iteração atual. Uma mutação na trilha de feromônio é introduzida para gerar diversidade no processo de busca. Além disso, esse modelo também considera a reinicialização da trilha de feromônio quando essa fica estagnada. O *Hyper-Cube AS* (Blum *et al.*, 2001) introduziu uma atualização do feromônio a partir de uma regra que define o peso de cada solução, escalonando os valores da função objetivo.

Os algoritmos ACOs são bem adequados para problemas NP-difíceis e de otimização estocástica em geral, tais como caminho mínimo (Dorigo & Gambardella, 1997), ordenação sequencial (Gambardella & Dorigo, 2000), atribuição quadrática (Gambardella *et al*., 1999), roteamento de veículos (Aloise *et al*., 2002; Bell & McMullen, 2004; Lopes *et al*., 2007) e de redes (Schoonderwoerd *et al*., 1996), escalonamento (Merkle & Middendorf, 2003), coloração de grafos (Costa & Hertz, 1997), otimização multiobjetivo (Chaharsooghi & Kermani, 2008), satisfação de restrições (Schoofs & Naudts, 2000) e cobertura de conjuntos (Lessing *et al*., 2004). O número de aplicações do ACO tem crescido fortemente, o que torna difícil citar exemplos em particular. Problemas industriais também têm recebido a contribuição dessa técnica, destacando-se o ajuste de parâmetros de controladores PID (Tan *et al*., 2005), problemas de manufatura (Solimanpur *et al*., 2004), de linhas de montagem (Ying & Liao, 2004) e de *layout* industrial (Hani *et al*., 2007), processamento de imagens (Nezamabadipour *et al*., 2006), roteamento em redes de telecomunicações (Di Caro & Dorigo, 1998), redes de sensores sem fio (Silva *et al*., 2007), despacho econômico de carga (Coelho *et al*., 2005, 2008), planejamento de rotas (Tavares Neto & Coelho, 2003, 2004, 2005), dentre vários outros. Áreas como bioinformática (Chan & Freitas, 2006) e composição de músicas (Geis & Middendorf, 2007) também têm sido exploradas com as técnicas de ACO.

Além das aplicações já mencionadas, essa metaheurística também vem sendo usada para o desenvolvimento de algoritmos de aprendizagem para estruturas de representação de conhecimento, como máquinas de vetores suporte (Martens *et al*., 2008), lógica nebulosa (Vieira *et al*., 2007) e redes Bayesianas (Correa *et al*., 2007) para tarefas relacionadas com mineração de dados (Parpinelli *et al*., 2002). Umas das tendências atuais das investigações sobre esse tema é a hibridização dos algoritmos ACO com os métodos mais clássicos de inteligência artificial ou de pesquisa operacional. Uma boa revisão dessa metaheurística pode ser encontrada em Dorigo & Blum (2005).

2.3 Algoritmo ACO Contínuo

Recentemente, pesquisas têm sido realizadas para estender os algoritmos de formiga para a solução de problemas de otimização no domínio contínuo. O primeiro método proposto foi o *Continuous ACO* (CACO), uma versão rudimentar proposta por Bilchev & Parmee (1995), e posteriormente consolidada em (Wodrich & Bilchev, 1997; Mathur *et al*., 2000).

Outros métodos desenvolvidos para a abordagem contínua incluem o algoritmo *Continuous Interacting Ant Colony* (CIAC) (Dréo & Siarry, 2002, 2004), o *Extended ACO* para domínios contínuos (ACO_R) (Socha, 2004; Socha & Dorigo, 2008), o *Continuous Ant Colony System* (CACS) (Pourtakdoust & Nobahari, 2004), o *Binary Ant System* (BAS) (Kong & Tian, 2005), o *Direct Application of ACO* (DACO) (Kong & Tian, 2006), o *Continuous Orthogonal Ant Colony* (COAC) (Hu *et al*., 2008) e outros algoritmos registrados em (Chen *et al*., 2003; Dréo & Siarry, 2006).

Versões no domínio contínuo do ACO têm sido utilizadas para tratar problemas como otimização de potência reativa em sistemas de energia elétrica (Lenin & Mohan, 2006), controle de processo químico para maximização da produção de metanol (Jalalinejad *et al*., 2007) e otimização da operação de reservatórios hídricos (Dariane & Moradi, 2008).

3 OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

O algoritmo de otimização por enxame de partículas (*Particle Swarm Optimization* – PSO) foi introduzido por James Kennedy e Russell Elberhart em 1995 para tratar problemas no domínio contínuo. O PSO emergiu de experiências com algoritmos que modelam o "comportamento social" observado em muitas espécies de pássaros e cardumes de peixes, e até mesmo do comportamento social humano.

Uma teoria sócio-cognitiva muito simples está por trás da PSO. Cada indivíduo de uma população possui sua própria experiência e é capaz de estimar a qualidade dessa experiência. Como os indivíduos são sociais, eles também possuem conhecimentos sobre como seus vizinhos comportam-se. Es-

ses dois tipos de informação correspondem à aprendizagem individual (cognitiva) e à transmissão cultural (social), respectivamente. Portanto, a probabilidade de que um determinado indivíduo tome uma certa decisão será uma função de seu desempenho no passado e do desempenho de alguns de seus vizinhos. Kennedy *et al*. (2001) utilizaram três princípios para resumir o processo de adaptação cultural:

- *Avaliar* – os indivíduos possuem a capacidade de sentir o ambiente de forma a estimar seu próprio comportamento;
- *Comparar* – os indivíduos usam uns aos outros como referência comparativa;
- *Imitar* – a imitação é central em organizações sociais humanas e é importante para a aquisição e manutenção das habilidades mentais.

Assim como as outras abordagens de inteligência coletiva, PSO está baseado em uma população de indivíduos capazes de interagir entre si e com o meio ambiente. Com base nas propriedades de autoavaliação, comparação e imitação, os indivíduos são capazes de lidar com um número de possíveis situações que o ambiente lhes apresenta. Os comportamentos globais serão, portanto, resultados emergentes dessas interações.

3.1 Algoritmo PSO

No algoritmo PSO, os indivíduos da população são representados por pontos, denominados de partículas, que voam em um espaço de busca $\Re^d$, onde d é a dimensão do espaço. As variações nos atributos desses pontos levam a novos pontos no espaço, ou seja, correspondem a movimentações no espaço. Uma ideia inspirada em sistemas cognitivos é a de que essas partículas tenderão a mover-se em direção umas às outras e irão influenciar umas às outras.

A maior parte dos algoritmos de PSO empregam dois princípios sócio-métricos, que representam dois tipos de informação importante no processo de decisão. O primeiro princípio (g_B) conecta conceitualmente todos os membros de uma população entre si. Como consequência, o comportamento de cada partícula é influenciado pelo comportamento de todas as outras partículas. A segunda métrica (p_B) cria uma vizinhança para cada indivíduo composta por ele próprio e seus vizinhos mais próximos. Ambas as métricas são medidas por uma função de avaliação ($f(p)$), também chamada função objetivo ou de aptidão (*fitness*), que corresponde à optimalidade da solução do problema.

Uma partícula p_i irá se mover em uma determinada direção que é função da posição atual da partícula $x_i(t)$, de uma velocidade $v_i(t$+1), da posição da partícula que levou ao seu melhor desempenho até o momento (p_B), e do melhor desempenho global do sistema até o momento (g_B). A velocidade da partícula será dada por:

$$v_i(t+1) = v_i(t)+\varphi_1\times(p_B-x_i(t))+\varphi_2\times(g_B-x_i(t)), \quad (4)$$

onde: φ_1 e φ_2 são constantes limitadas a um intervalo finito, em que Kennedy denomina-os como sendo respectivamente os componentes "cognitivo" e "social".

Uma vez que a velocidade da partícula é calculada, a posição da partícula i na próxima iteração é estabelecida como uma influência aditiva da posição antiga e da velocidade calculada, sendo expressa por:

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (5)$$

Para limitar a velocidade de uma partícula para que o sistema não extrapole o espaço de busca, são impostos limites ($\mathbf{v}_{max}$) para seus valores em cada dimensão (d) do espaço de busca:

$$\text{Se } v_i > v_{max} \text{ então } v_i = v_{max},$$
$$\text{Senão se } v_i < -v_{max} \text{ então } v_i = -v_{max}.$$

O algoritmo PSO é repetido até que um critério de terminação é atingido ou as mudanças nas velocidades das partículas estejam perto de zero. O pseudocódigo do algoritmo, em sua forma original, é descrito no Algoritmo 2.

Algoritmo 2: Pseudocódigo do PSO.

1. Determine o número de partículas P da população.
2. Inicialize aleatoriamente a posição inicial (x) de cada partícula p de P.
3. Atribua uma velocidade inicial (v) igual para todas as partículas.
4. Para cada partícula p em P faça:
   (a) Calcule sua aptidão *fp* = $f(p)$.
   (b) Calcule e melhor posição da partícula $p$até o momento (p_B).
5. Descubra a partícula com a melhor aptidão de toda a população (g_B).
6. Para cada partícula p em P faça:

(a) Atualize a velocidade da partícula pela fórmula:

$$v = v + \varphi_1 \times (p_B - -x) + \varphi_2 \times (g_B - -x).$$

(b) Atualize a posição da particular pela fórmula:

$$x = x + v.$$

7. Se condição de término não for alcançada, retorne ao passo 4.

3.2 Avanços e aplicações do algoritmo PSO

Para evitar os fenômenos de convergência prematura ou divergência que ocorriam com frequência no processo de otimização com o uso de PSO (especialmente em problemas complexos multidimensionais) e também para melhorar a velocidade de convergência e a precisão do PSO muitos tipos de esquemas foram introduzidos para realçar o PSO. Alguns dos esquemas mais representativos são: peso de inércia da velocidade (ω) (Shi & Eberhart, 1998); fator de constrição (χ), que um fator de amortecimento baseado nos parâmetros cognitivo (φ_1) e social (φ_2) para limitar a velocidade da partícula (Clerc, 1999; Eberhart & Shi, 2000); operação de cruzamento baseado nos mecanismos de reprodução dos algoritmos genéticos (Lovbjerg *et al*., 2001); EPSO, que é o uso de distribuição gaussiana dos parâmetros de inércia (ω), cognitivo (φ_1) e social (φ_2) (Miranda & Fonseca, 2002); autoadaptação (Lu & Hou, 2004); inclusão de um operador de evolução diferencial (DEPSO) na partícula no esquema anterior (Zhang & Xie, 2003); filtro de Kalman (KPSO) para atualização das posições das partículas (Monson & Seppi, 2004); CSV-PSO (Chen & Feng, 2005), que ajusta dinamica e não-linearmente o peso de inércia, o limite da velocidade de vôo e o espaço de vôo das partículas; vizinhança independente com subenxames independentes (INPSO) (Grosan *et al*., 2005); conceitos de seleção natural (DPSO) (Tillet *et al*., 2005); turbulência (TPSO) e turbulência adaptada nebulosamente (FATPSO) (Liu & Abraham, 2005); vetor de distribuição (VPSO), habilidade de cruzamento (COPSO) e adaptação ao entorno (LAPSO) (Yisu *et al*., 2007).

O algoritmo PSO modificado pela introdução do peso de inércia ou operação de cruzamento ou autoadaptação tem excelente capacidade de convergência com a diminuição da velocidade de convergência. O PSO com fator de constrição pode atingir o ótimo global rapidamente, mas o fenômeno de divergência ocorre nas soluções otimizadas. Divergência é quando a direção tomada pelas partículas leva a soluções distantes do ótimo global, sendo incapaz de retomar o caminho que levaria à solução ótima. PSO com subpopulação faz com que as partículas sobrevivam mais tempo. O CSV-PSO tem convergência mais rápida, maior probabilidade de convergência, além de ser mais estável. A turbulência utiliza um novo espaço de busca para superar o problema de estagnação da exploração das partículas, possibilitando que a partícula continue a mover-se e a manter a diversidade da população até a convergência do algoritmo.

Aplicações da técnica de PSO têm sido amplamente investigadas na literatura. Uma vertente é o uso do PSO em modelos híbridos para auxiliar a melhoria de desempenho de outros métodos e algoritmos de otimização, como algoritmos genéticos (Ru & Jianhua, 2008), redes neurais (Bashir & El-Hawari, 2009), sistemas nebulosos (Juang & Wang, 2009), evolução diferencial (Wickramasinghe & Li, 2008), *clustering* (Pei *et al*., 2008), etc. Outra linha é o uso do PSO na resolução de problemas em várias áreas de conhecimento. Em engenharia, uma ampla gama de aplicações, tais como: sintonização de controladores PID (Gaing, 2004; Falcucci *et al*., 2007), controles robustos (H_2/H∞) (Thanh & Parnichkun, 2008), identificação de modelos (Wang *et al*., 2008), otimização de controle de potência reativa e tensão (Abido, 2002), despacho econômico de carga (Wang & Singh, 2009), detecção de falhas (Tebaldi *et al*., 2006; Samanta & Nataraj, 2009), projeto de circuitos lógicos (Luna *et al*., 2004), robótica (Crispin, 2009), projeto de antenas (Wysota *et al*., 2008), otimização de redes de energia elétrica (Mo *et al*., 2007), despacho de carga (Abido, 2003), otimização de máquinas (Kashan & Karimi, 2009) e motores elétricos (Emara *et al*., 2008), predição (Zhao & Yang, 2009) e previsão (Izquierdo *et al*. 2009), jogos (Castro & Tsuzuli, 2008), processamento de imagens (Kwok *et al*., 2009), processamento de sinais (Xu & Gao, 2008), redes de sensores (Low *et al*., 2008), mineração de dados (Sousa *et al*., 2004), layout de facilidades (Müller *et al*., 2006), controle preditivo (Coelho & Krohling, 2003) e modelagem de sistemas dinâmicos (Araújo *et al*., 2009).

Uma revisão cuidadosa de princípios, variantes e aplicações do PSO pode ser encontrada em Banks *et al*. (2007, 2008), Poli *et al*. (2007) e del Valle *et al*. (2008).

3.3 Algoritmo PSO Discreto

Kennedy & Eberhart (1997) propuseram posteriormente uma versão binária de PSO que poderia atuar em funções bivaloradas. O propósito era facilitar a resolução de problemas combinatoriais, tais como escalonamento e roteamento, envolvidos na ordenação e arranjo de elementos discretos. Na adaptação realizada no algoritmo PSO, a posição de cada par-

tícula é descrita com 0 ou 1 em cada dimensão. A velocidade da partícula em uma dimensão particular representa a probabilidade da posição da partícula nessa dimensão ser 0 ou 1.

Trabalhos utilizando a abordagem discreta, incluindo às vezes pequenas modificações, têm sido aplicados a diversos problemas, tais como aproximação poligonal de curvas digitais (Yin, 2004), diagnóstico de máquinas (Guo *et al.*, 2006), escalonamento de tarefas em chão de fábrica (Liao *et al*., 2007), definição preliminar de configuração de aeronave (Blasi & Del Core, 2007) e projetos de circuitos lógicos combinacionais (Coello & Luna, 2003; Venayagamoorthy *et al*. 2007).

4 SHUFFLED FROG-LEAPING

O algoritmo *shuffled frog-leaping* (SFL) foi projetado por Eusuff (2003) como uma metaheurística para realizar uma busca heurística informada usando uma função heurística (uma função matemática qualquer) para procurar uma solução de um problema de otimização combinatorial. O SFL é fundamentado na abordagem baseada em população dos algoritmos meméticos, onde indivíduos interativos realizam troca global de informações entre eles.

O termo algoritmo memético foi cunhado por Dawkins (1976), originado de 'meme', que é um padrão de informação contagiosa que se replica infectando parasiticamente mentes humanas e/ou animais, alterando o seu comportamento e causando a propagação de um padrão (Eusuff *et al.*, 2006). O conteúdo atual de um meme, chamado 'memotipo', é análogo ao cromossomo de um gene. Uma ideia ou padrão de informação não é um meme até que alguém o transmita para outro ou até que um outro indivíduo o repita. Todo conhecimento transmitido é memético.

Dawkins (1976), em seu livro "*The Selfish Gene*", define o meme como simplesmente uma unidade de informação intelectual ou cultural que sobrevive tempo suficiente para ser reconhecida como tal e que pode passar de mente para mente. Exemplos de memes são músicas, ideias, gírias, roupas da moda e maneiras de fabricar panelas ou de construir arcos. Assim como os genes propagam-se em um *'pool'* de genes via esperma ou óvulos, os memos propagam-se no *'pool'* de memes passando de cérebro para cérebro via um processo que, em sentido amplo, pode ser chamado de imitação.

Os objetivos implícitos de genes e memes são diferentes por causa do uso de mecanismos distintos para passarem de um veículo para outro. Memes são positivamente selecionados principalmente para o aumento da comunicabilidade entre os hospedeiros (descritos como sapos no algoritmo SFL). Genes são selecionados principalmente por reprodução sexual.

As evoluções memética e genética estão sujeitas aos mesmos princípios básicos, ou seja, possíveis soluções são criadas, selecionadas de acordo com alguma medida de aptidão, combinadas com outras soluções e possivelmente sofrem mutação (Rahimi-Vahed & Mirzaei, 2007). A evolução memética, contudo, é um mecanismo muito mais flexível. Os genes somente podem ser transmitidos pelos pais, ou pelo pai no caso da reprodução assexuada, para seus descendentes. Os memes podem, em princípio, ser transmitidos entre dois indivíduos quaisquer. Visto que os genes são transmitidos entre gerações, organismos superiores podem precisar de vários anos para se propagar. Memes podem ser transmitidos no espaço de minutos. A replicação do gene é restrita pelo número bastante pequeno de descendentes de um único genitor, enquanto que o número de indivíduos que podem assumir um meme a partir de um único indivíduo é quase ilimitado. Além disso, parece muito mais fácil para memes suportar variação, uma vez que a informação no sistema nervoso é mais plástica que no DNA e que os indivíduos podem entrar em contato com muitas fontes diferentes de novos memes. Dessa forma, a difusão do meme é muito mais rápida que a do gene. A outra diferença entre memes e genes é que os memes são processados e possivelmente melhorados pelo indivíduo que os retém, e o que ocorre com os genes de um indivíduo.

4.1 Algoritmo SFL

O algoritmo *shuffled frog-leaping* representa um grupo de sapos saltando em um pântano. O pântano (espaço de busca) tem um número de pedras em localizações discretas (posições de possíveis soluções) nas quais os sapos (soluções) podem saltar para encontrar a pedra que possui a quantidade máxima de alimento disponível. Aos sapos é permitido comunicarem-se uns com os outros, de modo a melhorarem seus memes (ideias) usando a informação dos outros. A melhoria de um meme resulta em alterar a posição de um único sapo pela mudança no tamanho do salto (memotipo) do sapo.

O algoritmo progride pela transformação de sapos por meio de uma evolução memética. Nesse algoritmo, os sapos são vistos como hospedeiros de memes e descritos como um vetor memético. No SFL, a população consiste de um conjunto de sapos que é particionado em várias comunidades paralelas, que são subconjuntos chamados de memeplexes. Os diferentes memeplexes são considerados como diferentes culturas de sapos, localizados em diferentes sítios do pântano. Cada cultura de sapos (memeplex) realiza uma profunda busca local relativa a um objetivo. Dentro de cada memeplex, os sapos mantêm ideias individuais, que podem ser influenciadas pelas ideias dos outros sapos, e evoluem utilizando um processo de evolução memética (exploração local em um subconjunto do memeplex chamado submeme-

plex). Com um processo de infecção de ideias competitivo, os sapos com as melhores ideias contribuem mais para o desenvolvimento de novas ideias do que os sapos com ideias pobres. Durante a evolução, os sapos podem mudar seus memes usando a informação do melhor memeplex ou a melhor de toda a população (a avaliação da qualidade do meme é dada por uma heurística). Mudanças incrementais no memotipo correspondem a um tamanho do salto e o novo meme corresponde à nova posição do sapo. Depois que um sapo melhora sua posição, esse retorna para a comunidade (memeplex). A informação adquirida a partir de uma mudança na posição fica imediatamente disponível para ser melhorada mais adiante.

Após um número definido de passos evolutivos meméticos, as ideias são passadas entre os memeplexes em um processo de embaralhamento (*shuffling*). Tal processo realça a qualidade dos memes depois de infectados pelos sapos de diferentes regiões do pântano. A migração de sapos (fertilização cruzada de ideias) acelera o procedimento de busca compartilhando suas experiências na forma de infecção. Isso assegura que a evolução cultural relativa a algum interesse particular fica livre da tendência regional. A busca local e o processo de embaralhamento (realocação global) continuam até que os critérios de convergência do algoritmo sejam satisfeitos.

O Algoritmo 3, extraído de Eusuff *et al.* (2006) e sintetizado, descreve o procedimento usado para a resolução de um problema de otimização. Para maiores detalhes, consultar essa referência.

Algoritmo 3: Procedimento para o SFL

1. Determine o número de memeplexes (m) e o número de sapos (n) em cada memeplex e o número de sapos em um submemeplex (q). O tamanho total do pântano (F) é dado por $F = m \times n$.
2. Gere aleatoriamente a população virtual de F sapos.
3. Ordene os F sapos em ordem decrescente do valor de aptidão.
4. Registre a posição $\boldsymbol{P}_X$ do melhor sapo de toda a população (F).
5. Distribua n sapos em m memeplexes, de acordo com a aptidão.
6. Evolua cada memeplex,de acordo com o procedimento de exploração local a seguir:
   (a) Construa o submemeplex, selecionando aleatoriamente q sapos distintos dos n sapos do memeplex através de uma função de probabilidade triangular.
   (b) Ordene o submemeplex dos q sapos em ordem decrescente de aptidão.
   (c) Registre a melhor e a pior posição dos sapos no submemeplex como $\boldsymbol{P}_B$ e $\boldsymbol{P}_W$, respectivamente.
   (d) Melhore a posição $\boldsymbol{U}(q)$do pior sapo através das fórmulas:

$$S = \min\{int[rand() \times (P_B - P_W)], S_{max}\} \text{ (passo positivo)}, \quad S = \max\{int[rand() \times (P_B - P_W)], -S_{max}\} \text{ (passo negativo)}, \tag{6}$$

$$U(q) = P_W + S, \tag{7}$$

   onde: *rand* é um número aleatório no intervalo [0, 1] e $\boldsymbol{S}_{max}$ é o tamanho máximo do passo permitido para ser adotado por um sapo.
   (e) Calcule a nova aptidão. Se a posição do sapo não for melhorada ou é infactível dentro do espaço de busca, então o passo e a nova posição são calculados para esse sapo pelo tamanho do passo $\boldsymbol{S}$ e pela equação (8):

$$S = \min\{int[rand() \times (P_X - P_W)], S_{max}\} \text{ (passo positivo)}, \quad S = \max\{int[rand() \times (P_X - P_W)], -S_{max}\} \text{ (passo negativo)}, \tag{8}$$

   (f) Se a posição do sapo for melhorada, substitua a antiga posição e siga para o passo h.
   (g) Se a nova posição for infactível ou não for melhor que a posição antiga, gere aleatoriamente um novo sapo $r$em uma localização factível. Calcule a nova aptidão e substitua o sapo.
   (h) Atualize o memeplex. Após a alteração memética do pior sapo no submemeplex, substitua-o em sua localização original.
   (i) Ordene o memeplex em ordem decrescente do valor de aptidão.
7. Embaralhe os memeplexes evoluídos.
8. Ordene os F sapos em ordem decrescente do valor de aptidão.
9. Registre a posição $\boldsymbol{P}_X$ do melhor sapo de toda a população (F).
10. Se o critério de convergência for satisfeito, termine. Caso contrário, retorne à etapa 5.

4.2 Aplicações do algoritmo SFL

Existem poucos artigos dirigidos a aplicações do algoritmo SFL. Eusuff & Lansey (2003) usaram-no para determinar os tamanhos discretos ótimos de tubos para novas redes de tubulações e para expansão da rede de água. Liong & Atiquzzaman (2004) também desenvolveram um projeto de rede de distribuição de água com esse algoritmo. Eusuff *et al*. (2006) aplicaram-no ainda para resolver problema de otimização combinatorial.

Elbeltagi *et al*. (2005) compararam a formulação e os resultados de cinco algoritmos evolutivos: algoritmos genéticos, algoritmos meméticos, enxame de partículas, sistemas de colônia de formigas e o *shuffled frog-leaping*. Elbeltagi *et al*. (2007) usaram o algoritmo SFL para projetos de gerenciamento e engenharia de estrutura e infraestrutura.

Elbehairy *et al*. (2006) desenvolveram com este algoritmo um modelo para otimizar decisões de gerenciamento e consertos de pontes em rodovias. Rahimi-Vahed & Mirzaei (2007) trataram o problema de escalonamento *flow-shop* com permutação em um sistema de manufatura. Zhang *et al*. (2008) propuseram uma melhoria no algoritmo baseada em comportamento cognitivo para tratar problemas de otimização contínua e discreta. Sun *et al*. (2008) desenvolveram um método de classificação de documentos *web*. Huynh (2008) propôs uma modificação do algoritmo SFL para ajustar controladores PID.

5 OTIMIZAÇÃO POR CULTURA DE BACTÉRIA

De acordo com a lei de seleção natural, espécies com estratégias eficientes de coleta de alimentos sobrevivem, enquanto que aquelas com capacidades pouco eficientes de coleta são eliminadas ou transformadas em espécies melhores. Isto porque no primeiro caso a espécie está mais apta ao sucesso reprodutivo aumentando a concentração de espécies melhores nas futuras gerações. Tendo em vista que na busca por alimento um organismo ou animal toma ações para maximizar a energia utilizada por unidade de tempo gasta nessa tarefa e considerando todas as restrições presentes em sua própria fisiologia, tais como capacidades sensoriais, cognitivas e parâmetros ambientais ($e.g.$, densidade de presas, riscos de predadores, características físicas do local), a evolução natural pode conduzir à otimização do processo.

A atividade de busca por alimentos motivou pesquisadores ao uso deste princípio em processos de otimização. Em 2001, o Prof. Kevin Passino (Passino, 2001) propôs uma técnica de otimização conhecida como 'algoritmo de otimização pela coleta de alimentos por bactérias' (*Bacterial Foraging Optimization* – BFO). Este algoritmo foi baseado nas estratégias de localização, manipulação e ingestão de alimentos de células da bactéria *Escherichia Coli*, pois essa bactéria é um dos organismos mais conhecidos na atualidade.

Bactérias têm a tendência de migrar para áreas ricas em nutrientes utilizando um mecanismo chamado quimiotaxia. É sabido que a bactéria nada no meio líquido pela rotação dos flagelos, guiados por um propulsor reversível encaixado na parede da célula. A *E. coli* tem 8-10 flagelos colocados aleatoriamente no corpo celular. Quando todos os flagelos movem-se no sentido anti-horário, tornam-se compactos, impulsionando a célula ao longo de uma trajetória helicoidal. Quando os flagelos movem-se no sentido horário, todos eles puxam a bactéria em diferentes direções, o que resulta em uma mudança aleatória da direção de movimento, com deslocamento muito pequeno. Tal movimento (θ^i) justifica-se como uma forma de aumentar as chances da bactéria seguir um rumo que a faça atingir regiões mais favoráveis para obter nutrientes. Obviamente, a sequência de movimentos e a duração de um deslocamento unidirecional dependerão da concentração de nutrientes ou substâncias nocivas (ou ser neutra), ou ainda da presença de um gradiente nutritivo ou nocivo (dependente da variação espacial da densidade de nutrientes ou substâncias nocivas) na região onde a bactéria está.

O comportamento mótil padrão da bactéria em meio neutro é constituído por um revezamento entre deslocamentos contínuos (unidirecionais) e mudanças aleatórias de direção. Em um meio nutritivo homogêneo, há algumas alterações no comportamento mótil, como elevação da velocidade e duração dos deslocamentos contínuos e redução da duração das mudanças de direção. Considerando a presença de gradientes de nutrientes gerados pelo consumo de nutriente por um grupo de bactérias, as alterações podem ser ainda mais intensas. Em um gradiente nutritivo ascendente (ou nocivo descendente) uma bactéria irá elevar seu tempo de deslocamento, ao passo que em um meio descendente retornará ao seu comportamento padrão (busca de um gradiente nutritivo ascendente ou nocivo descendente).

Como a motilidade das células da *E. coli* agrupadas em colônias é formada pela agregação quimiotática, a população da *E. coli* varia de acordo com a quantidade de nutrientes disponível. Em condições favoráveis de crescimento, uma bactéria cresce suavemente em tamanho ou comprimento, e uma nova parede celular cresce pelo centro formando duas células-filhas, cada qual com o mesmo material genético da célula-mãe.

Em termos computacionais, o objetivo da BFO é implementar um percurso aleatório com uma certa tendência para cada bactéria, que deve dirigir-se para a concentração de nutrientes evitando substâncias nocivas para deixar o ambiente neutro o mais breve possível. O comportamento de busca de

nutrientes da *E. Coli* pode ser determinado pelos seguintes processos: (i) quimiotaxia, (*ii*) aglomeração, (*iii*) reprodução e (*iv*) eliminação e dispersão (Ali & Majhi, 2006).

A *quimiotaxia* determina o movimento de uma bactéria. A *E. Coli* pode mover-se de duas maneiras diferentes: nadar (*swim*) ou girar (*tumble*). A bactéria move-se em uma direção específica durante o nado. Durante o giro ela não tem uma direção certa de movimento e há pouco deslocamento. Geralmente a bactéria alterna entre estes dois modos de operação em durante todo o seu tempo de vida. Tal alternação entre os dois modos habilita a bactéria a mover-se em direções aleatórias e procurar por nutrientes.

Um passo quimiotático pode ser definido como um avanço (nado) seguido de um giro, ou um giro seguido de um avanço. Para representar um giro, uma unidade de comprimento do deslocamento aleatório $\phi(j)$ é gerada, e usada para definir a direção do movimento após um giro, representado por:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i) \times \phi(j), \qquad (9)$$

onde: $\theta^i(j,k,l)$ representa a posição da i-ésima bactéria na j-ésima época quimiotática, k-ésima época reprodutiva e l-ésima época de eliminação e dispersão. $C(i)$ é o tamanho do passo em certa direção aleatória especificada por um giro.

O giro é dado por um vetor aleatório $\Delta(i) \in \Re^d$, com cada elemento $\Delta_m(i)$, m = 1, 2, ..., d, dado por um número aleatório no intervalo [–1, 1], e d é o número de parâmetros a ser otimizado, de modo que:

$$\phi(j) = \frac{\Delta(i)}{\sqrt{\Delta^T(i) \times \Delta(i)}}. \qquad (10)$$

O custo (J) da bactéria i em uma certa localização (θ) é descrito por $J(i, j, k, l) = f(\theta^i(j,k,l))$, e representa uma determinada função objetivo (f) do problema tratado a ser minimizada, que por sua vez depende da posição (θ) da bactéria. A posição da bactéria representa uma solução potencial do problema.

Aglomeração descreve o processo de uma bactéria atrair outras bactérias, de modo que todas convirjam para a mesma posição. Para conseguir isso, uma função de penalidade baseada nas distâncias relativas de cada bactéria para a bactéria mais bem ajustada (mais saudável) é incluída na função de custo original. Finalmente, quando todas as bactérias fundem-se no ponto de solução, a função de penalidade torna-se zero.

As células de *E. coli* podem cooperativamente se auto-organizar em colônias altamente estruturadas com elevada adaptabilidade ambiental usando um intrincado mecanismo de comunicação ($e.g.$ sinalização quimiotática e troca plasmídica). De modo grosseiro, as células fornecem um sinal de atração para cada uma das outras, de modo que elas acabam por se agruparem. A representação matemática para a aglomeração, considerando a sinalização célula-a-célula via um atrator e um repulsor, pode ser representada por:

$$\begin{aligned} J_{cc}(\theta, P(j,k,l)) &= \sum_{i=1}^{S} J_{cc}^i(\theta, \theta^i(j,k,l)) \\ &= \sum_{i=1}^{S} -d_{\textit{atração}} \times \exp(-w_{\textit{atração}} \sum_{m=1}^{d} (\theta_m - \theta_m^i)^2) \ , \\ &+ \sum_{i=1}^{S} h_{\textit{repulsão}} \times \exp(-w_{\textit{repulsão}} \sum_{m=1}^{d} (\theta_m - \theta_m^i)^2) \end{aligned} \qquad (11)$$

onde: $J_{cc}(\theta,P(j,k,l))$ é um termo a ser adicionado à função de custo para apresentar uma função de custo variante no tempo; S é o número total de bactérias; d é o número de parâmetros a serem otimizados, presentes em cada bactéria; $d_{\textit{atração}}$, $w_{\textit{atração}}$, $h_{\textit{repulsão}}$ e $w_{\textit{repulsão}}$ são coeficientes diferentes que devem ser escolhidos adequadamente. $d_{\textit{atração}}$ é a profundidade do atrator liberado pela célula (uma quantificação de quanto atrator é liberado), $w_{\textit{atração}}$ é uma medida da largura do sinal atrator (uma quantificação da taxa de difusão química). A célula também repele uma célula próxima no sentido de que consome nutrientes próximos e não é fisicamente possível haver duas células na mesma posição, assim, $h_{\textit{repulsão}}$ é a altura do efeito repulsor (magnitude do seu efeito) e $w_{\textit{repulsão}}$ é uma medida da largura do sinal repulsor.

Um valor positivo de $J_{cc}(\theta,P(j,k,l))$ indica que a bactéria está em um ambiente rico em nutrientes, um valor igual a zero indica um ambiente neutro e um valor positivo indica um ambiente nocivo.

Assim, a função de custo da bactéria considerando a aglomeração pode ser dada por:

$$J(i,j,k,l) = J(i,j,k,l) + J_{cc}(\theta, P(j,k,l)). \qquad (12)$$

No processo de *reprodução*, após vários estágios de quimiotaxia, as bactérias são separadas em dois grupos de tamanhos iguais: as bactérias menos saudáveis morrem e as outras mais saudáveis dividem-se em duas na mesma localização. Com isso, assegura-se que a população de bactérias permaneça constante. A saúde da bactéria é calculada por:

$$J_{\textit{saúde}}^i = \sum_{j=1}^{N_c} J(i,j,k,l), \qquad (13)$$

onde: a saúde da bactéria é uma medida de quantos nutrientes conseguiu obter durante o seu tempo de vida e o quanto foi bem sucedida em evitar substâncias nocivas; e N_c é o número de passos quimiotáticos usados para a avaliação.

A *eliminação e dispersão* são eventos inesperados (com probabilidade de ocorrência igual a p_{ed}) que eliminam um conjunto de bactérias e/ou que as espalham em uma região não explorada do ambiente ou espaço de busca. Estes eventos têm o efeito de possivelmente destruir o progresso quimiotáxico, embora tenham também o efeito de colocar a bactéria mais próxima de uma nova fonte de nutrientes. Isso colabora na redução da probabilidade de cair em um mínimo local. Eliminação e dispersão são partes do comportamento móbil de longa distância a nível populacional.

5.1 Algoritmo BFO

As etapas envolvidas no procedimento de otimização baseado no comportamento de coleta de nutrientes por bactérias são mostradas no Algoritmo 4. Vale lembrar que as coordenadas de uma bactéria representam uma solução individual do problema de otimização. Um conjunto de soluções converge para uma solução ótima seguindo a dinâmica do grupo coletando alimentos na população de bactérias.

Algoritmo 4: Pseudocódigo do método de BFO

1. Inicialize o número de bactérias (S), o número de parâmetros a otimizar (d), o número de passos quimiotáticos (N_c) de cada bactéria i antes da reprodução, o número de movimentos (N_s) de cada bactéria da área de baixa para a área de alta concentração de nutrientes, o número de eventos de reprodução (N_{re}), número de eventos de eliminação e dispersão (N_{ed}), a probabilidade de eliminação e dispersão (p_{ed}), o vetor de direção do nado ($\phi(j)$), vetor de comprimento de deslocamento ($C(i)$), $d_{atração}$, $w_{atração}$, $h_{repulsão}$ e $w_{repulsão}$.
2. Coloque aleatoriamente cada bactéria i em uma posição inicial θ^i.
3. Início do laço (l) de eliminação e dispersão.
   - **3.1.** Início do laço (k) de reprodução.
     - **3.1.1.** Início do laço (j) de passos quimiotáticos.
       - **3.1.1.1.** Para cada bactéria i faça:
         (a) Calcule $J(i, j, k, l)$ incluindo J_{cc}.
         (b) $J_{ultimo} = J(i, j, k, l)$.
         (c) Calcule o giro $\phi(j)$ (*tumble*).
         (d) Realize o deslocamento $\theta^i(j+1, k, l)$ (*swim*).
         (e) Calcule o novo custo $J(i, j+1, k, l)$.
         (f) Início do loop (m) de movimento (nado).
         Se $J(i, j+1, k, 1) < J_{ultimo}$ faça:
         i. $J_{ultimo} = J(i, j+1, k, 1)$.
         ii. Realizar o deslocamento $\theta^i(j+1,k,l)$.
         iii. Calcular o novo custo $J(i, j+1, k, l)$.
         iv. Se $J(i, j+1, k, 1) \geq J_{ultimo}$ faça $m = N_s$
         v. Se m não atingir o fim do ciclo de nado N_s, retorne ao passo 3.1.1.1.f.i.
       - **3.1.1.2.** Se j não atingir o fim do ciclo de quimiotaxia N_c, retorne ao passo 3.1.1.1.
     - **3.1.2.** Calcule a saúde ($J^i_{saúde}$) de cada bactéria.
     - **3.1.3.** Ordene as bactérias de acordo com $J^i_{saúde}$.
     - **3.1.4.** Elimine as bactérias com os menores valores de $J^i_{saúde}$.
     - **3.1.5.** Duplique as bactérias com os maiores valores de $J^i_{saúde}$.
     - **3.1.6.** Se k não atingir o fim do ciclo de reprodução N_{re}, retorne ao passo 3.1.1.
   - **3.2** Para cada bactéria realize a eliminação e dispersão com probabilidade p_{ed}.
   - **3.3** Se l não atingir o fim do ciclo de eliminação e dispersão N_{ed}, retorne ao passo 3.1.

5.2 Avanços e aplicações do algoritmo BFO

Recentemente novos algoritmos de BFO têm sido desenvolvidos. O *Bacteria Chemotaxis* (BC) usa também o comportamento quimiotático das bactérias e foi desenvolvido por Muller *et al.* (2002) para problemas de otimização global. Biswas *et al.* (2007) desenvolveram um algoritmo híbrido que acopla características dos algoritmos BFO e PSO. Este algoritmo foi denominado *Bacterial Swarm Optimization* (BSO) e apresentou resultados bem promissores. Tang *et al.* (2007) propuseram um outro algoritmo, incorporando modelagens de estudos recentes sobre padrões de mobilidade de bactérias, denominado *Bacterial Swarming Algorithm* (BSA). Korani (2008) desenvolveu também um algoritmo híbrido usando BFO e PSO no mecanismo de eliminação e dispersão para a descoberta de novas soluções. O algoritmo foi denominado *Bacterial Foraging Optimization*

oriented by Particle Swarm Optimization (BF-PSO) e aplicado inicialmente ao problema de sintonia de controladores PID.

O algoritmo BFO tem sido aplicado com sucesso na sintonia de controladores adaptativos (Passino, 2002), na otimização de funções (Liu & Passino, 2002), no ajuste dos ganhos de controlador PID (Kim *et al*., 2005; Ali & Majhi, 2006; Niu *et al*., 2006; Yijian & Yanjun, 2008), no controle de manipulador robótico (Coelho & Silveira, 2006), na identificação de sistemas dinâmicos não-lineares (Majhi & Panda, 2007), estimação de sinal (Acharya *et al*., 2007), na redução da perda de transmissão em sistemas de potência (Mishra, 2007; Mishra *et al*., 2007), em problemas dinâmicos de otimização de custos em sistemas de potência (Tang *et al*., 2008), na estimação de parâmetros (Dang *et al*., 2008), no controle de antenas (Guney & Basbug, 2008a-b), no despacho econômico de carga (Saber & Venayagamoorthy, 2008), no processamento de imagens (Maitra & Chatterjee, 2008).

Gazi & Passino (2003, 2004) estudaram a estabilidade de colônias de bactérias. Kim *et al*. (2007) propuseram uma abordagem híbrida envolvendo algoritmos genéticos (AG) e BFO para otimizações de funções. O algoritmo híbrido superou o AG e o BFO em algumas referências comparativas numéricas e em um problema prático de projeto de sintonia de controladores PID. Chen *et al*. (2008) apresentaram uma variação do algoritmo BFO original através da introdução de abordagens cooperativas, obtendo uma melhora significativa em suas velocidade de convergência, precisão e robustez.

6 COLÔNIA DE ABELHAS

O comportamento de coleta de alimentos e características de aprendizagem, memorização e compartilhamento de informações têm sido umas das mais interessantes áreas em inteligência de enxames. As abelhas estão entre os insetos sociais mais estudados, devido à sua alta capacidade organizacional. Por causa disso, nos últimos anos, a quantidade de algoritmos desenvolvidos baseados em colônias de abelhas tem aumentado bastante. Diferentemente de outras técnicas de inteligência coletiva, onde uma metáfora e uma metaheurística sobre um certo tipo de colônia foram desenvolvidas e, na sequência, uma série de algoritmos foram agregados, os algoritmos baseados no comportamento de abelhas não seguiram esse mesmo caminho. Diversas abordagens relativas a diferentes características comportamentais da abelha foram propostas, produzindo algoritmos distintos.

Lucic & Teodorovic (2001) foram os primeiros pesquisadores a empregar os princípios básicos da inteligência coletiva das abelhas na solução de problemas de otimização combinatorial, introduzindo um algoritmo chamado *Bee System* (BS), que foi usado para resolver o problema do caixeiro viajante. Abbas (2001a-b) desenvolveu um algoritmo de otimização baseado no processo de acasalamento das abelhas. Sung (2003) introduziu um algoritmo de evolução de abelha-rainha para realçar a capacidade otimização de algoritmos genéticos, conhecido como *Queen-bee Evolution Algorithm* (QEGA). Nakrani & Tovey (2003) desenvolveram o *Honey Bee Algorithm* (HBA) para realizar alocação de servidores na Internet. Wedde *et al*. (2004) apresentaram um novo algoritmo de roteamento chamado *BeeHive*, que foi inspirado na comunicação e métodos e procedimentos de avaliação de abelhas produtoras de mel. Bozorg Haddad & Ashfar (2004) desenvolveram um algoritmo baseado no vôo nupcial das abelhas, conhecido como *Mating Bee Optimization* (MBO) aplicado ao problema de otimização de reservatório usando variáveis discretas, que foi posteriormente melhorado para otimização de funções contínuas (Bozorg Haddad *et al*., 2006), chamada *Honey-Bee Mating Optimization* (HBMO).

Teodorovic & Dell'Orco (2005) propuseram o *Bee Colony Optimization Metaheuristic* (BCO), capaz de resolver tanto problemas combinatoriais determinísticos quanto problemas combinatoriais caracterizados por incerteza. Yang (2005) desenvolveu o *Virtual Bee Algorithm* (VBA) para otimização de funções contínuas de duas variáveis. Drias *et al*. (2005) introduziram uma nova abordagem inteligente ou nova metaheurística chamada *Bees Swarm Optimization* (BSO), que é inspirada no comportamento de abelhas reais. Os autores adaptaram-no para as características do problema de satisfabilidade máxima ponderada (max-sat problem). Benatchba *et al*. (2005) propuseram uma solução para o problema 3-sat baseado no processo de reprodução das abelhas.

Pham *et al*. (2006a) desenvolveram um algoritmo de otimização inspirado no comportamento de coleta de alimentos de abelhas produtoras de mel, chamado *Bees Algorithm* (BA). Basturk & Karaboga (2006) descreveram um algoritmo de colônia de abelhas artificiais na coleta de alimentos (*Artificial Bee Colony* – ABC).

Lu & Zhou (2008) desenvolveram um algoritmo para convergência global inspirados no comportamento de coleta de pólen das abelhas (*Bee Collecting Pollen Algorithm* – BCPA). Outros algoritmos que introduzem modificações nestes algoritmos de abelha citados também foram propostos e novos algoritmos estão atualmente em desenvolvimento (Baig & Rashid, 2007; Li *et al*., 2008; Lemmens *et al*., 2008).

Fazendo uma inspeção cuidadosa na literatura, os algoritmos de colônias de abelhas podem ser categorizados basicamente em três linhas (Baykasoglu *et al*., 2007):

- *Comportamentos de coleta de alimentos* (Lucic & Teodorovic, 2001; Wedde *et al.*, 2004; Yang, 2005; Teodorovic & Dell'Orco, 2005; Drias *et al*, 2005; Karaboga &

Basturk, 2007; Pham *et al*., 2006a; Lu & Zhou, 2008);

- *Comportamentos de acasalamento* (Abbas, 2001a-b; Bozorg-Haddad & Afshar, 2004; Benatchba *et al*., 2005; Bozorg-Haddad *et al*., 2006; Chang, 2006; Afshar *et al*., 2007); e
- *Conceito de abelha-rainha* (Sung, 2003; Kara, 2004; Azeem & Saad, 2004).

6.1 Algoritmos de colônias de abelhas

Em função dos diferentes mecanismos empregados na implementação dos algoritmos de colônias de abelhas, a seguir será apresentado um algoritmo de cada categoria comportamental, com uma breve introdução dos conceitos biológicos que nortearam a construção dos modelos computacionais.

6.1.1 Comportamento de coleta de alimentos

A auto-organização das abelhas está baseada em poucas regras simples do comportamento individual do inseto. Um exemplo é a coleta e o processamento do néctar, práticas altamente organizadas, que incluem mecanismos de comunicação e compartilhamento de informações eficientes. Entre as abelhas melíferas (produtoras de mel), a comunicação pode ser feita por meio de sons, substâncias químicas, tato, danças ou estímulos eletromagnéticos. A transferência de alimento parece ser uma das maneiras mais importantes de comunicação. A dança é outro importante meio de comunicação; por meio dela as operárias podem informar a distância e a localização exata de uma fonte de alimento.

As abelhas são dotadas de processo de orientação excepcional, que é baseado principalmente na posição do sol, isto é, o ângulo entre sua própria rota de vôo e uma linha horizontal da colmeia, na direção do sol (bússola solar), registrando uma posição de que jamais se esquecem. Trata-se de uma espécie de memória geográfica. Além disso, elas podem se guiar também pela cor e odor das flores.

Uma colônia de abelhas deve acumular comida suficiente no verão e consumi-la durante o inverno, quando as fontes de alimento tornam-se escassas. A coleta de alimentos é feita pelas abelhas campeiras (operárias). A colônia de abelhas coordena sua atividade de coleta de uma maneira eficiente, enviando as abelhas em múltiplas direções simultaneamente para explorar uma grande área de busca. A colônia de abelhas concentra, então, a busca nas fontes de néctar mais proveitosas dentre todas da florada (Bahamish *et al*., 2008).

Depois que colhe o néctar, a abelha volta à colmeia, mas se lembra do odor, cor e forma das flores. Dentro do favo e, portanto, longe da luz solar, a abelha campeira baila de maneiras distintas conforme a distância da florada. Cada colmeia tem uma assim chamada área de pista de dança na qual as abelhas que descobriram fontes de néctar dançam, de modo a tentar convencer suas companheiras a segui-las. Cada abelha seguidora decide atingir a fonte de néctar ou pólem seguindo uma abelha dançarina que já descobriu uma florada. Durante o vôo, a abelha campeira pega uma carga de néctar e retorna à colmeia transferindo o néctar para uma abelha operária que armazena-o. Depois de ceder o alimento, a abelha pode: (a) abandonar a fonte de alimento e tornar-se novamente uma exploradora, (b) continuar a coleta na fonte de alimento sem recrutar as companheiras, ou (c) dançar e assim recrutar as companheiras antes de retornar para a fonte de alimento. As abelhas optam por uma das alternativas acima com uma certa probabilidade. Dentro da área de dança, a abelha dança 'anunciando' diferentes áreas de néctar. Durante a dança, a abelha campeira indica a direção da fonte de alimento em relação à posição da colmeia e do sol. A campeira pode interromper sua dança a curtos intervalos e oferecer às abelhas seguidoras que estão observando-a, uma gota do néctar que coletou. Assim, ela informa o odor do néctar e da flor e as demais operárias partem em busca desta fonte. O recrutamento aumenta com a vivacidade e a duração da dança. O tipo de dança que a abelha realiza depende da distância da colmeia à florada. A distância da colmeia até a fonte de néctar é informada pelo número de vibrações (requebrados) realizadas e pela intensidade do som emitido durante a dança. Quanto menor a distância entre a fonte e a colmeia, maior o número de vibrações. Portanto, as danças indicam que há uma fonte abundante de alimento, os movimentos indicam a distância e a orientação, e o néctar ou pólem passado para as recrutas ajudam a reconhecer a fonte pelo odor do alimento.

Os mecanismos pelos quais a abelha decide seguir uma dançarina específica não são bem conhecidos, mas considera-se que o recrutamento entre as abelhas é sempre uma função da qualidade da fonte de alimento. Sabe-se também que nem todas as abelhas começam a buscar alimento simultaneamente.

Embora um número bastante significativo de algoritmos de abelhas utilize esta abordagem como modelo, apenas um deles é aqui descrito, de forma bastante simplificada e reduzida, para fins de exemplificação. O algoritmo escolhido é o *Artificial Bee Colony* (ABC). Informações mais específicas e detalhadas sobre esta e outras implementações devem ser buscadas em suas referências bibliográficas.

Nesse algoritmo a posição de uma fonte de alimento representa uma solução possível do problema de otimização e a quantidade de néctar corresponde à qualidade ou medida de aptidão da solução associada (equivalente ao valor da função-objetivo) (Karaboga, 2005). A colônia de abelhas artificiais consiste de três grupos de abelhas operárias: campeiras, seguidoras e escudeiras. As abelhas campeiras estão associa-

das com uma fonte de alimento particular, a qual estão explorando. Elas carregam a informação sobre essa fonte particular e compartilham essa informação com uma certa probabilidade pela dança. A abelha seguidora espera na pista de dança para tomar a decisão de qual fonte de alimento escolher para explorar. Quando a quantidade de néctar de uma fonte de alimento aumenta, o valor da probabilidade com o qual a fonte de alimento é preferida pelas seguidoras também aumenta. A primeira metade da colônia consiste de abelhas campeiras e a segunda metade de seguidoras. Para toda fonte de alimento existe uma abelha campeira, ou seja, o número de abelhas campeiras é igual ao número de fontes de alimento em torno da colmeia. A abelha campeira – cuja fonte de alimento foi exaurida pelas abelhas – torna-se uma abelha escudeira, cujo papel é explorar o ambiente, sem nenhuma direção, para descobrir novas fontes de alimento. Como resultado desse comportamento, as escudeiras são caracterizadas por baixos custos de busca e baixa média na qualidade da fonte de alimento. Ocasionalmente, as abelhas escudeiras podem acidentalmente descobrir ricas fontes de alimento inteiramente desconhecidas. O Algoritmo 5 apresenta o pseudocódigo[1] ABC.

Algoritmo 5: Procedimento ABC.

1. Determine o tamanho da colônia de abelhas (*COL*), o número inicial de abelhas campeiras (*BN*); o número de fontes de alimento (*SN*), que é igual a *BN*; o número inicial de abelhas seguidoras (*BC*), que é igual à diferença entre *COL* e *BN*; o número de abelhas escudeiras (*BE*); e o número de tentativas de liberar uma fonte de alimento (*lim*).

2. Envie aleatoriamente as abelhas campeiras para as fontes de alimento iniciais (x_i), no espaço D-dimensional.

3. Envie as abelhas seguidoras para as melhores fontes de alimento encontradas pelas campeiras e determine as quantidades de néctar (*fit*(x_i)) (aptidão) coletadas por cada uma.

4. Calcule o valor de probabilidade (P) das fontes (n) que serão escolhidas pelas abelhas campeiras (i), usando a expressão:

$$P_i = \frac{fit(x_i)}{\sum_{n=1}^{SN} fit(x_n)}. \qquad (14)$$

5. Interrompa o processo de explotação das fontes abandonadas pelas abelhas (piores fontes).

6. Envie as escudeiras, aleatoriamente, para a área de busca para descobrir novas fontes de alimento na vizinhança, onde a posição (v) das novas fontes (i) é dada por:

$$v_{ij} = x_{ij} + \phi_{ij} \times (x_{ij} - -x_{kj}), \qquad (15)$$

onde: $k \in$\{1, 2, ..., *BN*\}e $j \in$\{1, 2, ..., D\} são índices escolhidos aleatoriamente, de modo que $k \neq i$. ϕ_{ij} é um número aleatório entre [-1, 1].

7. Memorize a melhor fonte de alimento encontrada até o momento.

8. Se o número de tentativas (*nt*) de descobrir novas fontes de alimento fracassar (*nt > lim)*, ou seja, se durante *lim* tentativas as fontes de alimento não melhorarem então as abelhas escudeiras devem abandonar suas fontes estagnadas e buscar aleatoriamente novas fontes de alimento (x_i) no espaço D-dimensional.

9. Se condição de término não for alcançada, retorne ao passo 3.

[1] O pseudo-código do algoritmo ABC pode ser encontrado em http://mf.erciyes.edu.tr/abc/

6.1.2 Comportamento de acasalamento de abelhas

Uma colônia de abelhas típica consiste de uma única abelha-rainha por vez, nenhum ou milhares de zangões, e entre 10.000 e 60.000 operárias (Abbas, 2001a). A rainha, durante o seu tempo de vida, tem a função de colocar ovos e reproduzir a espécie. Os zangões têm a função de fecundar a rainha. As larvas que nascem de ovos fertilizados podem tornar-se rainhas ou operárias, enquanto que os não-fertilizados tornam-se zangões. A função das abelhas operárias é coletar néctar e cuidar da prole, por exemplo, alimentando novas rainhas (realeiras) com geleia real produzida por abelhas nutrizes. A dieta com geleia real torna as rainhas maiores e mais fortes que qualquer outra abelha da colmeia. Como em uma colmeia só pode haver uma rainha, a primeira a nascer luta com outras rainhas que tenham nascido no mesmo período até que apenas uma sobreviva. Se a nova rainha for mais forte que a antiga, elimina ou expulsa-a juntamente com seu séquito de algumas centenas de abelhas, de modo que o enxame sai em busca de um local propício para construir nova colmeia.

No processo de acasalamento, a rainha sai da colmeia seguida por um séquito formado por todos os zangões adultos e voa para o alto, cada vez mais alto. Apenas os zangões mais rápidos, aptos e fortes, portanto capazes de gerar a prole mais saudável, conseguem alcançá-la e o acasalamento ocorre em pleno vôo. No acasalamento, o esperma do zangão atinge um repositório chamado "espermateca" e fica acumulado para formar uma matriz genética para a colônia. Com o término da inseminação, o zangão morre. A rainha pode acasalar com vários zangões. Após o vôo nupcial a rainha jamais deixará a colmeia novamente. Ela terá acumulado em sua espermateca uma quantidade de sêmen de zangões suficiente para, ao longo de sua vida, fecundar milhares de ovos. Toda vez que a rainha puser ovos, ela aleatoriamente recuperará uma mistura acumulada na espermateca para fertilizar os ovos.

Em uma colônia de abelhas artificial, o vôo nupcial pode ser considerado um conjunto de transições em um espaço de estados (ambiente), em que a rainha move-se entre os diferentes estados em uma certa velocidade, podendo acasalar com o zangão de acordo com uma certa probabilidade. No início do vôo, a rainha começa com um certo conteúdo de energia e retorna para a sua colmeia quando a energia está próxima a zero ou a espermateca está cheia. As abelhas-rainhas na colmeia artificial representam as soluções potenciais de um problema, enquanto que o papel da abelha operária está restrito ao cuidado com a prole (soluções candidatas) e pode ser interpretado através de uma heurística ou um operador genético, como a mutação, atuando para melhorar o conjunto de crias.

O procedimento *Honey-Bee Mating Optimization* (HBMO), usado principalmente para buscar soluções de problemas de otimização global com variáveis discretas ou contínuas, pode ser construído, de maneira simplificada, como apresentado no Algoritmo 6.

Algoritmo 6: Pseudocódigo do procedimento HBMO.

1. Defina o número de operárias, o número de zangões, o tamanho da espermateca (número de acasalamentos de uma rainha), o número total de crias (capacidade de postura) da rainha, a velocidade da rainha no início (*Smax*) e no final do vôo de acasalamento (*Smin*), a taxa de redução da velocidade de vôo (α = [0, 1]), a quantidade de energia da rainha (E), e o fator de redução da quantidade de energia da rainha (γ).

2. Gere aleatoriamente a população inicial de zangões e de operárias (formação da colmeia).

3. Calcule a função-objetivo (aptidão) (f) da população.

4. Ordene a população com base na aptidão.

5. Selecione a rainha (Q), que é a operária com a melhor aptidão.

6. Determine aleatoriamente a velocidade (S) e a energia (E) do vôo da rainha:

$$S(0) = rand() \times (Smax - -Smin) + Smin, \quad (16)$$

$$E(0) = E, \quad (17)$$

onde: *rand*() é um número entre [0, 1].

7. Para o acasalamento da rainha faça:

   (a) Comece o vôo nupcial, selecionando probabilisticamente um zangão (D) para formar a espermateca (lista de zangões), de acordo com a função:

$$Prob(D) = \exp(-\Delta(f)/S(t)), \quad (18)$$

   onde: $\Delta(f)$ é a diferença absoluta entre a aptidão do zangão $f(D)$ e a aptidão da rainha $f(Q)$.

   (b) Selecione aleatoriamente um número (*r* = *rand*()) para verificar se o zangão entra na lista para a inseminação. Se este número for menor que *Prob*(D) então ordene o esperma do zangão na espermateca da rainha.

   (c) Decremente a velocidade do vôo e a energia da rainha pelas fórmulas (20) e (21).

$$S(t+1) = \alpha(t) \times S(t), \quad (19)$$

$$E(t+1) = E(t) - -\gamma. \quad (20)$$

   (d) Se*r* > *Prod*(D) retorne ao passo 3.a

   (e) Enquanto *S*> *Smin* e E > 0 e espermateca não estiver cheia, retorne ao passo 3.a.

8. Gere novas crias (C) (soluções de teste) empregando operadores genéticos de cruzamento (ou heurísticas) pré-definidos entre os genótipos dos zangões escolhidos e os da rainha.

9. Operárias alimentam as crias e a rainha com geleia real na proporção de suas aptidões. Use diferentes heurísticas ou o operador de mutação para melhorar a prole.

10. Atualize a aptidão das crias, baseado nas melhorias obtidas.

11. Substitua a rainha mais fraca pela cria mais forte, de modo que a rainha não possa ser mais fraca que uma cria. Elimine as crias restantes.

12. Gere novos zangões aleatoriamente (Eles morrem no acasalamento ou abandonam a colmeia forçados pelas operárias).

13. Se o critério de terminação não for atingido retorne ao passo 6.

Para maiores detalhes deste algoritmo, como equações e heurísticas, consultar Bozorg Haddad & Ashfar (2004) e Ashfar *et al*. (2007). Uma versão para *clustering* desse algoritmo pode ser encontrada em Fathian & Amiri (2008).

6.1.3 Evolução da abelha-rainha

Como mencionado anteriormente, a rainha é a figura central e a responsável pela manutenção populacional (reprodução da espécie) de uma colônia. A fecundação da rainha ocorre em áreas de congregação de zangões, onde existem de centenas a milhares de zangões voando à espera de uma rainha, conferindo assim uma grande variabilidade genética no acasalamento. Por ser responsável pela transmissão de toda a informação genética à família, as características dos indivíduos da colmeia serão diretamente dependentes da qualidade da abelha-rainha. Em casos especiais, as operárias também podem produzir ovos, embora não fertilizados, que darão origem a zangões.

Quando uma rainha diminui a quantidade de ovos, ou seja, a sua capacidade de postura, as operárias responsáveis pela manutenção das larvas promovem o desenvolvimento de nova rainha para continuar o desenvolvimento da colmeia.

O algoritmo de evolução da abelha-rainha (QEGA) foi proposto por Sung (2003) para melhorar a capacidade de otimização dos algoritmos genéticos (Goldberg, 1989), e possui duas diferenças principais em comparação com o este último. Primeiro, os pais $P(t)$ no algoritmo genético são compostos por um número n de indivíduos gerados por uma seleção como a roleta, por exemplo, enquanto que os pais $P(t)$ na evolução da abelha-rainha consistem de n/2 parceiros (I_m) de uma abelha-rainha $I_q(t–1)$, onde q = arg max $\{f_i(t–1), 1 \leq i \leq n\}$, e cada abelha escolhida I_m (t–1), com $1 \leq m \leq n/2$, é determinada segundo um algoritmo de seleção.

Segundo, todos os indivíduos no algoritmo genético sofrem mutação com uma pequena probabilidade p_m, enquanto que na evolução da abelha-rainha apenas parte dos indivíduos sofrem mutação com a probabilidade normal de mutação p_m e os outros sobrem mutação com probabilidade de mutação forte p'_m. A razão entre p_m e p'_m é dado com ξ no algoritmo. Geralmente, p_m é menor que 0,1 e p'_m é maior que p_m.

A primeira característica da evolução da abelha-rainha realiza o refinamento das soluções existentes, ainda que a descendência dependa principalmente da operação de *crossover* e da aptidão individual. Isso aumenta a probabilidade da convergência prematura. A segunda característica ajuda os algoritmos genéticos a explorar novas regiões do espaço de busca.

Estas duas características habilitam os algoritmos genéticos a evoluir rapidamente, assim como manter boas soluções. Por fim, a evolução da abelha-rainha torna possível esses algoritmos atingirem rapidamente o ótimo global, diminuindo a probabilidade de convergência prematura. O procedimento QEGA é apresentado no Algoritmo 7.

Algoritmo 7: Pseudocódigo do QEGA.

1. Determine o tamanho da populaçãoP, a razão de mutação normal ξ, a probabilidade normal de mutação p_m, a probabilidade normal de mutação p'_m.

2. Inicialize aleatoriamente os indivíduos da população P.

3. Avalie a aptidão da população P.

4. Selecione $P(t)$ a partir de $P(t–1)$, encontrando a abelha-rainha I_q e as abelhas-selecionadas I_m, ou seja:

$$P(t) = \{ (I_q (t–1), I_m(t–1)) \}.$$

5. Recombine $P(t)$ fazendo a operação de cruzamento.

6. Para cada indivíduo i em P faça mutação. **Se** $i \leq (\xi \times n)$ então faça mutação com probabilidade p_m **senão** faça mutação com probabilidade p'_m.

7. Avalie a aptidão da população P.

8. Se a condição de término não for alcançada, retorne ao passo 4.

6.2 Aplicações dos algoritmos de abelhas

Embora os algoritmos baseados em colônias de abelhas sejam relativamente recentes, algumas aplicações têm sido propostas na literatura, tais como a solução do problema do caixeiro viajante (Lucic & Teodorovic, 2003; Wong *et al*., 2008), determinção de estratégias de aterrisagem em veículos aéreos não-tripulados (Chahl *et al*., 2004), despacho econômico de energia (Qin *et al.*, 2004), sintonização de controladores (Azeem & Saad, 2004; Leivislö & Joensuu, 2006; Pham *et al*., 2007b; Serapião, 2009), escalonamento *job shop* (Chong *et al.*, 2006), otimização de pesos de redes neurais (Pham *et al*., 2006b,c,d), otimização de máquinas de vetores suportes (Pham *et al*., 2007e), direção de robôs (Curkovic & Jerbic, 2007), projeto de células de manufatura (Pham *et al*., 2007a), projeto de junção de soldas (Pham *et al*., 2007d), particionamento e escalonamento (Koudil *et al*., 2007; Banerjee *et al*. 2008), alocação de recursos (Quijano & Passino, 2007), problema de atribuição (Baykasoglu *et al.*, 2007), planejamento de transporte (Wedde, *et al*., 2007), calibração de modelos hidrológicos (Barros *et al*., 2007), determinação de conformação de proteínas (Bahamish *et al.*, 2008), codificação de DNA (Xu *et al*., 2008), problema de roteamento de veículos (Marinakis *et al*., 2008b) e de locações (Marinakis *et al*., 2008b), análise de *clusters* (Fathian & Amiri, 2008), extração de regras (Bozorg Haddad *et al*., 2008), aprendizagem de cinemática inversa de robôs (Pham *et al*., 2008), projeto de filtros digitais (Karaboga, 2009).

O desenvolvimento de algoritmos de abelhas híbridos com outras técnicas computacionais, tais como PSO (Yang *et al*., 2007a), *wolf pack search* (Yang *et al*., 2007b), cadeias de Markov (Meng *et al*., 2006; Yang *et al*., 2007c), algoritmo k-médias (Fathian *et al.*, 2007), também vem sendo investigado.

Ainda assim, a quantidade de artigos demonstrando a aplicabilidade dos algoritmos de abelha é de certa forma limitada. A abordagem tem sido biologicamente bastante investigada e tem demonstrado ser uma técnica muito promissora, despertando interesse da comunidade científica.

7 EXPERIMENTOS

Para uma melhor compreensão do potencial dos algoritmos apresentados foram realizados dois estudos de caso. O primeiro estudo está voltado para o problema de otimização global para minimização de funções. O segundo caso é o uso desses métodos como ferramenta de aplicação para a solução do problema de despacho econômico de carga em sistemas de energia.

Os estudo foram realizados com os algoritmos PSO, SFL, BFO e ABC. Todos os algoritmos foram implementados de acordo com os procedimentos descritos nas suas versões originais, que são as mesmas reproduzidas na parte teórica deste trabalho. Não foram considerados refinamentos ou modificações surgidas em estudos posteriores, embora essas possam acarretar em melhorias nos resultados obtidos nos experimentos. A escolha das versões originais ocorreu em respeito à preservação da intenção inicial de cada autor no desenvolvimento das metaheurísticas aqui abordadas.

Os métodos foram todos implementados no Matlab 7.4.0 (MathWorks). Os programas foram executados em uma máquina com processador Intel Core 2 Duo de 1,83 GHz e 2 GB de RAM.

7.1 Estudo de caso 1: Otimização funcional

Para avaliar o desempenho dos algoritmos apresentados na otimização de funções, algumas funções de teste clássicas foram utilizadas: função F6 de Schaffer (f_1), da esfera (f_2), de Alpine (f_3), de Rastrigin (f_4), de Rosenbrock (f_5) e de Griewank (f_6). Conforme mostrado na Tabela 1, nos experimentos o número de variáveis utilizadas para cada função é indicado através da dimensão e os limites do espaço de busca estabelecidos como o mesmo para todas as dimensões da função. Os valores de mínimo global e da solução ótima de cada função também são descritos. Todas as funções possuem mínimo global em 0. As funções de Schaffer, da esfera, de Alpine e de Rastrigin atingem o mínimo global quando todas as variáveis assumem valor 0. Na função de Rosenbrock, quando todas as variáveis possuem o valor 1, a função encontra seu valor mínimo, ao passo que para Griewank o mesmo é obtido quando todos os parâmetros são iguais a 100.

A função F6 de Schaffer é bidimensional, apresentando vários mínimos locais. A função da esfera é contínua, convexa e unimodal. As funções de Alpine e Rastrigin são multimodais e apresentam vários mínimos locais. A função de Rosenbrock possui um ótimo global dentro de um longo vale parabólico e a convergência para este ponto é uma tarefa difícil, pois as variáveis são fortemente dependentes, embora o ótimo global seja único e a função unimodal. A função de Griewank é multimodal, mas em uma alta dimensionalidade (n>30), pode ser vista como unimodal.

Os valores dos parâmetros de cada algoritmo adotados nas simulações são mostrados na Tabela 2. A nomenclatura das variáveis segue a indicada nos algoritmos, como explicado nas seções anteriores.

Em todos os métodos o número de avaliações (critério de convergência) foi estabelecido em 3.000 iterações para as funções f_1, f_2 e f_3, porém para as funções f_4, f_5 e f_6

Tabela 1: Funções numéricas de teste.

Nome da função	Função	Dimensão	Limites	Valor mínimo
F6 de Schaffer	$f_1(\vec{x}) = 0,5 + \frac{\text{sen}^2(\sqrt{x_1^2 + x_2^2}) - 0,5}{(1 + 0,001(x_1^2 + x_2^2))^2}$	2	[-100; 100]	$f_1(\vec{0}) = 0$
Esfera	$f_2(\vec{x}) = \sum_{i=1}^{n} x_i^2$	5	[-100; 100]	$f_2(\vec{0}) = 0$
Alpine	$f_3(\vec{x}) = \sum_{i=1}^{n} \text{abs}(x_i \text{sen}(x_i) + 0,1x_i)$	10	[-5,12; 5,12]	$f_3(\vec{0}) = 0$
Rastrigin	$f_4(\vec{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	50	[-5,12; 5,12]	$f_4(\vec{0}) = 0$
Rosenbrock	$f_5(\vec{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	50	[-50; 50]	$f_1(\vec{1}) = 0$
Griewank	$f_6(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{n}(x_i - 100)^2\right) - \left(\prod_{i=1}^{n}\left(\frac{x_i - 100}{\sqrt{i}}\right)\right) + 1$	50	[-600; 600]	$f_6(1\vec{00}) = 0$

Tabela 2: Valores dos parâmetros usados pelos algoritmos de otimização.

PSO		SFL		BFO		ABC	
P (partículas)	20	m (memeplexes)	10	S (bactérias)	20	COL (colônia	20
φ_1 (cognitivo)	2	n (sapos)	10	N_c	10	BN (campeiras)	10
φ_2 (social)	2	q (submeme-plexes)	2	N_S	4	BC (seguidoras)	10
		S_{max} (passo)	5	N_{re}	4	$Cmax$	100
		iN (evoluções)	5	N_{ed}	iterações		
				p_{ed}	$0,25$		
				$d_{\text{atração}}$	$0,1$		
				$w_{\text{atração}}$	$0,2$		
				$h_{\text{repulsão}}$	$0,1$		
				$w_{\text{repulsão}}$	10		

considerou-se 5.000 iterações. No método BFO o número de ciclos de eliminação e dispersão (N_{ed}) foi colocado como sendo igual ao número de iterações para produzir o mesmo efeito dos outros algoritmos.

7.2 Estudo de caso 2: Despacho econômico de carga

O objetivo básico do problema de despacho econômico da geração de energia termoelétrica é o escalonamento das saí-

das das unidades de geração conveniadas para encontrar a demanda de carga consumidora a um custo mínimo de operação, satisfazendo as restrições inerentes às unidades geradoras utilizadas e as restrições de igualdade e desigualdade impostas pelo problema.

O despacho econômico para a operação de grupos térmicos é descrito por um problema de programação matemática multiobjetivo que consiste em minimizar a função que determina o custo de combustível (função objetivo), encontrando um perfil de geração ótimo, sujeito à satisfação da carga de energia elétrica e aos limites técnicos de operação dos grupos.

Considere um parque de grupos térmicos com n unidades geradoras. O custo total de combustível na geração de energia a ser minimizado é a soma das contribuições das unidades geradoras dada por:

$$\min \sum_{i=1}^{n} F_i(P_i), \tag{21}$$

onde: F_i é a função custo de combustível para a unidade geradora i(em \$/h) e P_i (em MW) é a potência fornecida por essa unidade.

O custo variável de operação de cada unidade poder ser expresso em função da potência de saída:

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i, \tag{22}$$

sujeito a:

$$P_i^{\min} \leq P_i \leq P_i^{\max}, \tag{23}$$

onde: a_i, b_i e c_i são restrições das características do gerador, e P_i^{min} e P_i^{max} são respectivamente as saídas de operação mínimas e máximas da unidade geradora i (em MW).

A energia gerada a partir de todas as unidades comissionadas deve satisfazer à demanda de carga, que é definida como:

$$\sum_{i=1}^{n} P_i - P_D - P_L = 0, \tag{24}$$

onde: P_D é a demanda de carga total do sistema (em MW); P_L é a perda de transmissão (em MW).

No despacho econômico de carga, as perdas de transmissão do sistema devem calculadas como funções das saídas das unidades geradoras usando a matriz de coeficiente B, da seguinte forma:

$$P_L = \sum_{i}^{n} \sum_{j}^{n} B_{ij} P_i P_j + \sum_{i}^{n} B_{i0} P_i + B_{00}, \tag{25}$$

onde: B_{ij} é o *ij*-ésimo elemento da matriz de coeficiente de perda, B_{i0} é o i-ésimo elemento do vetor de coeficiente de perda e B_{00} é a constante do coeficiente de perda.

A equação (23) representa as restrições de desigualdade relativas aos limites da capacidade de geração de potência de cada unidade geradora, enquanto que a equação (24) representa as restrições de igualdade do balanço de potência (ou seja, equilíbrio entre suprimento e demanda).

Os algoritmos de otimização baseados em colônias foram utilizados para minimizar a função objetivo abaixo:

$$\min(f) = \sum_{i=1}^{n} F_i(P_i) + q_1 \left(abs \left(\sum_{i=1}^{n} P_i - P_D - P_L \right) \right), \tag{26}$$

onde: q_1 é uma constante positiva para penalizar as soluções que não atendem ao equilíbrio no balanço de carga.

Esta função objetivo foi estabelecida para contemplar não apenas a distribuição de carga nas unidades geradoras com custo baixo como também para satisfazer a restrição de igualdade do sistema.

Dois cenários distintos foram investigados no estudo de caso do despacho econômico de carga em um sistema termoelétrico. Os parâmetros utilizados nos algoritmos para o tratamento deste problema foram os mesmos indicados na Tabela 2. Em cada simulação realizou-se 5000 iterações.

7.2.1 Simulação 1: Sistema com três unidades geradoras

O primeiro cenário de investigação compreende três unidades geradoras. A demanda de potência (P_D) a ser encontrada pelas unidades geradoras é 150 MW. Os dados do sistema são apresentados na Tabela 3.

Para este sistema, as matrizes B dos coeficientes de perda na linha de transmissão (com capacidade de base de 100 MW) são dadas respectivamente por:

$$B_{ij} = 10^{-2} \times \begin{bmatrix} 0,0218 & 0,0093 & 0,0028 \\ 0,0093 & 0,0228 & 0,0017 \\ 0,0028 & 0,0017 & 0,0179 \end{bmatrix},$$

$$B_{i0} = 10^{-3} \times \begin{bmatrix} 0,3 & 3,1 & 1,5 \end{bmatrix},$$

$$B_{00} = 0,030523.$$

Tabela 3: Dados para as três unidades termais de geração sobre capacidade e coeficientes.

Unidade	a ($/MW^2)	b ($/MW)	c ($)	P^{min} (MW)	P^{max} (MW)
1	0,008	7	200	10	85
2	0,009	6,3	180	10	80
3	0,007	6,8	140	10	70

7.2.2 Simulação 2: Sistema com seis unidades geradoras

No segundo cenário, o sistema consiste de seis unidades geradoras, com 26 barramentos e 46 linhas de transmissão. O sistema fornece uma demanda de carga total (P_D) de 700 MW. Em relação ao caso 1, a complexidade e a não-linearidade do problema são aumentadas. Na Tabela 4 estão indicados os dados do sistema.

Tabela 4: Dados para as seis unidades termais de geração sobre capacidade e coeficientes.

Unidade	a ($/MW^2)	b ($/MW)	c ($)	P^{min} (MW)	P^{max} (MW)
1	0,007	7	240	100	500
2	0,0095	10	200	50	200
3	0,009	8,5	220	80	300
4	0,009	11	200	50	150
5	0,008	10.5	220	50	200
6	0,0075	12	120	50	120

As matrizes B dos coeficientes de perda na linha de transmissão (com capacidade de base de 100 MW) são dadas respectivamente por:

$$B_{ij} = 10^{-4} \times \begin{bmatrix} 0{,}14 & 0{,}17 & 0{,}15 & 0,19 & 0,26 & 0,22 \\ 0,17 & 0,60 & 0,13 & 0,16 & 0,15 & 0,20 \\ 0,15 & 0,13 & 0,65 & 0,17 & 0,24 & 0,19 \\ 0,19 & 0,16 & 0,17 & 0,71 & 0,30 & 0,25 \\ 0,26 & 0,15 & 0,24 & 0,30 & 0,69 & 0,32 \\ 0,22 & 0,20 & 0,19 & 0,25 & 0,32 & 0,85 \end{bmatrix},$$

$$B_{i0} = 10^{-3} \times [-0{,}3908 \;\; -0,1297 \;\; 0,7047 \;\; 0,0591 \;\; 0,2161 \;\; -0,6635]$$

$$B_{00} = 0,056.$$

8 RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados dos experimentos computacionais conduzidos para avaliar o desempenho dos algoritmos apresentados. Os resultados de cada estudo de caso são indicados separadamente e algumas considerações gerais sobre os parâmetros que influenciam o desempenho dos algoritmos são tecidas posteriormente, incluindo também alguns comentários sobre o tempo de computação de cada técnica.

8.1 Resultados do estudo de otimização funcional

Cada um dos experimentos foi repetido 20 vezes, independentemente, com diferentes valores iniciais da população aleatoriamente escolhidos. A Tabela 5 compara os algoritmos pela qualidade das soluções ótimas. Os valores das soluções ótimas encontrados em cada execução de cada função de teste foram armazenados e a média e o desvio padrão (σ) dessas funções foram calculados para cada algoritmo. Os valores menores que 10^{-10} foram registrados como 0. As melhores soluções de cada caso são marcadas em negrito.

A Tabela 5 mostra que o algoritmo ABC apresentou as melhores médias na obtenção do ótimo das funções, incluindo os menores desvios padrões em todas as execuções. O ABC for pior apenas que o PSO no caso da função de Alpine (f_3). A média do algoritmo SFL é menor que o BFO em todos os casos, com exceção da função esfera (f_2). Em relação ao PSO a média é menor somente nos casos de alta dimensionalidade (f_4, f_5 e f_6). O PSO mostrou resultados muito bons em relação a todos os outros algoritmos nas funções com baixa dimensionalidade. O BFO apresentou o pior desempenho em relação a todos os algoritmos, incluindo os maiores desvios padrão.

Os resultados desse estudo de caso mostram que o algoritmo ABC é robusto, flexível e simples, e pode ser usado eficientemente na otimização de problemas multimodais e multivariáveis, pois produziu resultados significantemente melhores que os outros algoritmos. Resultados semelhantes podem ser encontrados em (Karaboga & Basturk, 2007).

8.2 Resultados do estudo do problema de despacho econômico de carga

Para investigar os efeitos das soluções iniciais no resultado final, o problema foi resolvido com 20 soluções iniciais diferentes, aleatoriamente determinadas. Os resultados obtidos para ambas as simulações em termos de melhor custo, pior custo, custo médio e desvio padrão das execuções são reportados na Tabela 6. Os melhores resultados de cada quesito foram destacados em negrito.

Tabela 5: Resultados obtidos pelos algoritmos PSO, SFL, BFO e ABC para otimização de funções de teste.

Função	PSO		SFL		BFO		ABC	
	Média	σ	Média	σ	Média	σ	Média	σ
f6 (Schaffer) (f_1)	0,0128	0,0312	**0**	0	0,0767	0,0259	**0**	0
Esfera (f_2)	**0**	0	0,413	0,728	0,0344	0,0268	**0**	0
Alpine (f_3)	**5,6E-8**	1,5E-7	0,075	0,096	1,2131	0,5139	6,4E-5	7,2E-5
Rastrigin (f_4)	279,042	38,582	235,139	47,981	317,334	28,724	**0**	0
Rosenbrock (f_5)	729,656	441,107	21676,5	20668,7	226,432	132,164	**0,0043**	0,0044
Griewank (f_6)	58,960	52,168	2,402	0,549	287,139	28,748	**0**	0

Tabela 6: Resultados obtidos pelos algoritmos PSO, SFL, BFO e ABC para minimização do custo de despacho de carga em 20 execuções.

	Simulação 1				**Simulação 2**			
Método	Custo médio ($/h)	Desvio padrão	Custo máximo ($/h)	Custo mínimo ($/h)	Custo médio ($/h)	Desvio padrão	Custo máximo ($/h)	Custo mínimo ($/h)
PSO	1609,13	8,231	1627,87	1600,60	8722,04	177,652	8912,16	8401,45
SFL	**1602,06**	**1,519**	**1607,62**	1600,67	8479,49	**54,781**	**8604,29**	8419,78
BFO	1604,28	3,1993	1611,35	**1600,02**	8571,91	127,85	8909,85	8428,69
ABC	1607,37	11,676	1620,60	1600,51	**8457,16**	57,726	8610,28	**8372,27**

Tabela 7: Resultados da melhor simulação para o problema de despacho de carga para o sistema com três unidades termais (simulação 1).

Saídas de potência	**PSO**	**SFL**	**BFO**	**ABC**
Unidade 1 (MW)	30,617	30,85	33,435	33,049
Unidade 2 (MW)	66,759	62,488	65,450	61,764
Unidade 3 (MW)	55,385	59,357	53,806	57,872
P_L (MW)	2,76	2,70	2,75	2,70
P_D (MW)	150	150	150	150
ΣP_i (MW)	152,76	152,69	152,69	152,69
Custo ($)	1600,60	1600,67	1600,02	1600,51

Valores baixos de desvio padrão indicam uma boa convergência dos métodos. Nesta perspectiva o SFL superou os demais métodos nos dois testes, mostrando-se mais robusto. Inclusive apresentando também o menor custo máximo nos dois testes, assim como o menor custo médio na simulação 1, e o segundo lugar na simulação 2. Apesar disso, em nenhuma situação foi o método capaz de atingir o menor custo mínimo

Tabela 8: Resultados da melhor simulação para o problema de despacho de carga para o sistema com seis unidades termais (simulação 2).

Saídas de potência	PSO	SFL	BFO	ABC
Unidade 1 (MW)	288,653	287,392	222,260	323,043
Unidade 2 (MW)	82,753	67,637	58,777	54,965
Unidade 3 (MW)	132,988	140,933	150,395	147,354
Unidade 4 (MW)	50,00	98,357	106,963	50,00
Unidade 5 (MW)	99,565	64,052	101,601	85,815
Unidade 6 (MW)	57,768	53,150	72,559	50,233
P_L (MW)	11,73	11,59	11,73	11,40
P_D (MW)	700	700	700	700
ΣP_i (MW)	711,73	711,52	711,39	711,40
Custo ($)	8401,45	8419,78	8428,69	8372,27

O algoritmo ABC expressou os menores custos médio e mínimo e o segundo menor desvio padrão e custo máximo para a simulação 2. Esta abordagem parece superar os demais métodos com o aumento da dimensionalidade, reforçando o percebido no primeiro estudo de caso.

O PSO foi superado por todos os outros métodos na avaliação de custo médio e custo máximo. Obteve o pior desempenho quando comparado o desvio padrão na simulação 2 e o segundo pior na simulação 1. Ou seja, este algoritmo apresentou pior taxa de convergência e menor robustez.

O BFO apresentou o segundo melhor desempenho na simulação 1, mas com o aumento da dimensionalidade na simulação 2, seu desempenho não foi tão expressivo.

As Tabelas 7 e 8 indicam o desempenho de cada algoritmo na obtenção do mais baixo custo mínimo dentre as 20 execuções. Pode-se notar que embora as potências de cada unidade encontradas pelos algoritmos sejam diferentes, elas produziram resultados semelhantes no tocante à potência total do sistema ($\sum P_i$), à perda de carga na transmissão (P_L) e ao custo total na primeira simulação, pois guardam uma certa relação entre si. Isso sugere a possibilidade da existência de uma tendência de soluções ótima para esse caso. Na simulação 2 os resultados são bem distintos. Quando se comparam as melhores soluções para esse caso, que são as fornecidas pelos algoritmos PSO e ABC, e também as piores soluções, fornecidas pelo SFL e BFO, não é possível identificar com clareza uma propensão de valores ótimos ou de valores que prejudicam o critério de minimização de custo.

8.3 Algumas considerações

Os experimentos mostraram que os resultados dos algoritmos são fortemente influenciados pelas condições inicias geradas aleatoriamente para a população. O fator que comprova essa afirmação é o desvio padrão encontrado nos estudos.

Embora não tenham sido avaliados no presente estudo, o tamanho da população, os limites do espaço de busca, o número de iterações e a alteração dos valores dos parâmetros inerentes a cada algoritmo afetam diretamente a qualidade das soluções. Assim, sendo os resultados obtidos em qualquer método podem ser melhorados (ou piorados) modificando-se os parâmetros. De modo geral, mostrou-se que os algoritmos tendem pelo menos a levar as soluções para regiões na proximidade de um ótimo global. O ajuste dos parâmetros auxilia no aumento da precisão (refinamento) da solução.

Até agora não parece ter sido demonstrado que um método em particular possa ser usado como o método mais eficiente de propósito geral para a resolução de problemas quaisquer. Ao que parece, alguns métodos apenas se adaptam melhor a algumas classes de problemas que outros, sob certas condições.

Os algoritmos PSO e ABC são de mais fácil implementação e menor complexidade computacional, portanto, são mais rápidos que o SFL e o BFO em tempo de execução. Em aplicações de tempo real, esta pode ser uma diferença crucial. A Tabela 9 indica os tempos de computação (em segundos) exigidos para uma única execução da busca de solução do primeiro estudo de caso para a minimização da função de Rosenbrock (f_5) e para a solução do primeiro exemplo de despacho econômico de carga (termoelétrica 1) do segundo

estudo de caso. Ambos os casos consideraram 5000 iterações.

Tabela 9: Tempo de uma única execução dos algoritmos para dos exemplos dos estudos de caso.

Método	*Rosenbrock* (s)	*Termoelétrica 1* (s)
PSO	24,761	9,615
SFL	391,110	514,249
BFO	678,917	593,038
ABC	10,807	14,978

Vale lembrar que as metaheurísticas investigadas foram recentemente propostas e estão abertas para receber contribuições que podem corrigir eventuais insuficiências em algum sentido. Também é importante mencionar que tais metaheurísticas, embora inspiradas em comportamentos de sistemas biológicos, não têm como principal foco a modelagem de processos da natureza, e sim tratar problemas de otimização através da exploração de espaços de busca de forma não-analítica.

De acordo com de Castro (2006), os métodos de computação natural são indicados em otimização quando:

- O problema a ser resolvido é complexo, por exemplo, envolve um grande número de variáveis ou soluções potenciais, é altamente dinâmico, não-linear, tem múltiplos objetivos, etc.
- Não é possível garantir que a solução encontrada é ótima, mas é possível encontrar uma medida de qualidade que permita a comparação entre soluções.
- Quando uma única solução não é suficiente adequada ou quando diversidade é importante, já que os métodos determinísticos apontam uma única solução.

9 CONCLUSÕES

Nos últimos anos as técnicas de inteligência de enxames têm experimentado um crescimento, tanto no tocante à multiplicidade de algoritmos em desenvolvimento quanto na variedade das aplicações resultantes. Este crescimento é devido, em parte, a fatores como o grande número de problemas que envolvem características como não-linearidade, alta dimensionalidade, dificuldade de modelagem, impossibilidade no cálculo de derivadas, entre outros. Essas dificuldades tornam infactível o uso de técnicas tradicionais, tais como programação combinatória (Korte & Vygen, 2008), linear (Chvátal, 1983), não-linear (Luenberger, 2003), dinâmica (Bellman, 2003), inteira (Wolsey, 1998), etc.

Os algoritmos de enxame, de modo geral, têm mostrado resultados extremamente promissores em termos de robustez de descoberta de soluções, da velocidade da solução e precisão numérica. Grande parte desse êxito é também devido à simplicidade, versatilidade, generalidade e paralelismo desses algoritmos, sobretudo quando comparados à dificuldade de modelagem matemática de certos problemas.

Por outro lado, esses métodos ainda enfrentam dificuldades na aplicação em larga escala de problemas de otimização de alta dimensão, principalmente por causa da carga computacional que impõem. Contudo, com o desenvolvimento do poder computacional das máquinas modernas e da interação de pesquisadores multi e interdisciplinares engajados no aprimoramento dos algoritmos e no estudo e desenvolvimento de novos modelos, o futuro das técnicas baseadas em colônias parece ser bem auspicioso e instigante.

As aplicações na área de engenharia de tais métodos são as mais diversas, desde robótica, controle automático, roteamento, solução de problemas combinatoriais, problemas de otimização, redes de comunicação, energia e hidráulica, até sistemas de manufatura, processamento de imagens, bioinformática, identificação de sistemas e classificação de dados, dentre outras.

O propósito deste trabalho não foi esgotar o assunto ou indicar o estado da arte dos métodos de inteligência coletiva aqui expostos, e nem identificar o melhor método heurístico de resolução de problemas, mas sim reunir algumas informações relevantes e fornecer subsídios para possíveis utilizações dos algoritmos no tratamento de problemas de engenharia. Sendo assim, a principal contribuição foi esclarecer e diferenciar as abordagens baseadas em comportamentos de sistemas naturais para a otimização de problemas em geral, sugerindo uma enorme gama de potenciais temas de investigação.

REFERÊNCIAS

Abbass, H.A. (2001a). Marriage in honey-bee optimization (MBO): a haplometrosis polygynous swarming approach. *Proceedings of the 2001 Congress on Evolutionary Computation*, pp. 207–214.

Abbass, H.A. (2001b). A monogenous MBO approach to satisfiability. *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation*.

Abido M. (2002). Optimal design of power system stabilizers using particle swarm optimization. *IEEE Trans Energy Conversion*, Vol. 17, No. 3, pp. 406–413.

Abido, M. A. (2003). Environmental/economic power dispatch using multiobjective evolutionary algorithms.

IEEE Transactions on Power Systems, Vol. 18, No. 4, pp. 1529–1537.

Acharya, D.P.; Panda, G.; Mishra, S.; Lakshmi, Y.V.S. (2008). Bacteria foraging based independent component analysis. *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications* (ICCIMA 2007), paper no. 4426753, pp. 527–531.

Afshar, A.; Bozorg Haddad, O.; Mariño, M.A.; Adams, B.J. (2007). Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. *Journal of the Franklin Institute*, Vol. 344, pp. 452–462.

Ali, A.; Majhi, S. (2006). Design of Optimum PID Controller by Bacterial Foraging Strategy. *Proceedings of the IEEE International Conference on Industrial Technology (ICIT 2006)*, pp. 601–605.

Aloise, D.; Maia, R.S.; Aloise, D.J.; Ochi, L.S.; Bittencourt, V.G. (2002). Uma Colônia de Formigas para o Problema de Extração de Petróleo e Otimização de Rotas de Unidades Móveis de Pistoneio. *Anais do XIV Congresso Brasileiro de Automática* (CBA), pp. 1232–1237.

Araújo, E.; Araújo, F.P.A.; Sakita, M.T.; Figueiroa, J.G.S.; Souza, H.A.F. (2009). Dynamical Vibration System Modeling by using Particle Swarm Optimization with Turbulence. *Proceedings of the XX International Congress of Mechanical Engineering* (COBEM), Gramado, RS.

Azeem M.F.; Saad A.M. (2004). Modified Queen Bee Evolution Based Genetic Algorithm for Tuning of Scaling Factors of Fuzzy Knowledge Base Controller. *IEEE INDICON 2004 Proceedings of the India Annual Conference*, pp. 299–303.

Bahamish, H.A.A.; Abdullah, R.; Salam, R.A. (2008). Protein Conformational Search Using Bees Algorithm. *Proceedings of the* 2^{th} *Asia International Conference on Modelling & Simulation*, pp. 911–916.

Baig, A.R.; Rashid, M. (2007). Honey bee foraging algorithm for multimodal & dynamic optimization problems. *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference*.

Banerjee, S.; Dangayach, G.S.; Mukherjee, S.K.; Mohanti, P.K. (2008). Modelling process and supply chain scheduling using hybrid meta-heuristics. *Studies in Computational Intelligence*, Vol. 128, pp. 277–300.

Banks, A.; Vincent, J.; Anyakoha, C. (2007). A review of particle swarm optimization. Part I: Background and development. *Natural Computing*, Vol. 6, No. 4, pp. 467–484.

Banks, A.; Vincent, J.; Anyakoha, C. (2008). A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, Vol. 7, No. 1, pp. 109–124.

Barros, F.V.F.; Martins, E.S.P.R.; Nascimento, L.S.V.; Reis Jr., D.S. (2007). Calibração de Modelos Hidrológicos Utilizando Algoritmos Evolucionários Multiobjetivo. *Anais do XVII Simpósio Brasileiro de Recursos Hídricos e* 8^o *Simpósio de Hidráulica e Recursos Hídricos dos Países de Língua Oficial Portuguesa*, São Paulo, SP.

Bashir, Z.A.; El-Hawary, M.E. (2009). Applying Wavelets to Short-Term Load Forecasting Using PSO-Based Neural Networks. *IEEE Transactions on Power Systems*. Article in Press.

Basturk B.; Karaboga D. (2006), An Artificial Bee Colony (ABC) Algorithm for Numeric Function Optimization, *IEEE Swarm Intelligence Symposium*, Indianapolis, Indiana, USA.

Baykasoglu, A.; Özbakır, L.; Tapkan, P. (2007). Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem. In: *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, F.T.S. Chan and M.K. Tiwari (Eds.), Itech Education and Publishing, Vienna, Austria, pp. 113–144.

Bell, J.E.; McMullen, P.R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, Vol. 18, pp. 41–48.

Bellman, R. (2003). *Dynamic Programming*. Dover Publications.

Benatchba, K.; Admane, L.; Koudil, M. (2005). Using bees to solve a data-mining problem expressed as a max-sat one, artificial intelligence and knowledge engineering applications: a bioinspired approach. *Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation*.

Bilchev, G.; Parmee, I.C. (1995). The ant colony metaphor for searching continuous design spaces. *Proceedings of the AISB Workshop on Evolutionary Computation*, University of Shepheld, UK, LNCS 933, Springer-Verlag, pp. 25–39.

Biswas, A.; Dasgupta1, S., Das, S.; Abraham, A. (2007). Synergy of PSO and Bacterial Foraging Optimization - A Comparative Study on Numerical Benchmarks. In:

Innovations in Hybrid Intelligent Systems, E. Corchado *et al*. (Eds.), ASC 44, Springer Verlag, pp. 255–263.

Blasi, L.; Del Core, G. (2007). Particle Swarm Approach in Finding Optimum Aircraft Configuration. *Journal of Aircraft*, Vol. 44, No. 2, pp. 679–683.

Blum, C.; Roli, A.; Dorigo, M. (2001). HC–ACO: The hyper-cube framework for Ant Colony Optimization. *Proceedings of the Metaheuristics International Conference*, Vol. 2, pp. 399–403.

Bozorg Haddad O., Afshar A. (2004). MBO Algorithm, A New Heuristic Approach in Hydrosystems Design and Operation. *Proceedings of the 1^{st} International Conference on Managing Rivers in the 21^{st} Century*, pp. 499–504.

Bozorg Haddad O.; Afshar A.; Mariño M.A. (2006). Honey-Bees Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization. *Water Resources Management*, Vol. 20, pp. 661–680.

Bozorg Haddad, O.; Afshar, A.; Mariño, M.A. (2008). Honey-bee mating optimization (HBMO) algorithm in deriving optimal operation rules for reservoirs. *Journal of Hydroinformatics*, Vol. 10, No. 3, pp. 257–264.

Bullnheimer, B.; Hartl, R.F.; Strauss, C. (1997). A new rank based version of the Ant System — a computational study. Institute of Management Science, University of Vienna, *Technical Report*.

Castro, E.G.; Tsuzuki, M.S.G. (2008). Swarm Intelligence applied in synthesis of hunting strategies in a three-dimensional environment. *Expert Systems with Applications*, Vol. 34, No. 3, pp. 1995–2003.

Chaharsooghi, S.K.; Kermani, A.H.M. (2008). An intelligent multi-colony multi-objective ant colony optimization (ACO) for the 0-1 knapsack problem. *Proceedings of the IEEE Congress on Evolutionary Computation* (CEC 2008), pp. 1195–1202.

Chahl, J.S.; Srinivasan, M.V.; Zhang, S.W. (2004). Landing Strategies in Honeybees and Applications to Uninhabited Airborne Vehicles. *The International Journal of Robotics Research*, Vol. 23, No. 2, pp. 101–110.

Chan, A.; Freitas, A.A. (2006). A new ant colony algorithm for multi-label classification with applications in bioinfomatics. *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (GECCO '06), pp. 27–34.

Chang H.S. (2006). Converging Marriage in Honey-Bees Optimization and Application to Stochastic Dynamic Programming. *Journal of Global Optimization*, Vol. 35, pp. 423–441.

Chen, B.R; Feng, X.T. (2005). Particle swarm optimization with contracted ranges of both search space and velocity. *Journal of Northeastern University (Natural Science)*, Vol. 26, No. 5, pp. 488–491.

Chen, H.; Zhu, Y.; Hu, K.; He, X.; Niu, B. (2008). Cooperative approaches to bacterial foraging optimization. *Lecture Notes in Artificial Intelligence*, Vol. 5227, pp. 541–548.

Chen, L.; Shen, J.; Qin, L.; Chen, H.J. (2003). An improved ant colony algorithm in continuous optimization. *Journal of Systems Science and Systems Engineering*, Vol. 12, No. 2, pp. 224–235.

Chong C.S.; Low M.Y.H.; Sivakumar A.I.; Gay K.L. (2006). A Bee Colony Optimization Algorithm to Job Shop Scheduling. *Proceedings of the 37^{th} Winter Simulation*, Monterrey, California, pp. 1954–1961.

Chvátal. V. (1983). *Linear Programming*. W. H. Freeman.

Clerc, M. (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Proceedings of the 1999 Congress Evolutionary Computation*, IEEE Press, pp. 1951–1957.

Coelho, L.S.; Krohling, R. (2003). Controlador preditivo baseado em otimização por colônia de partículas. *Anais do VI Congresso Brasileiro de Redes Neurais* (CBRN), São Paulo.

Coelho, L.S.; Mariani, V.C. (2005). Abordagem de colônia de formigas híbrida aplicada à otimização do despacho econômico de energia elétrica. *Anais do VII Simpósio Brasileiro de Automação Inteligente* (SBAI), São Luis, Maranhão.

Coelho, L.S; Sierakowski, C.A.; Mariani, V.C. (2008). Algoritmo inspirado em colônia de formigas e método de busca local aplicado à otimização de despacho econômico de energia elétrica: fundamentos e um estudo de caso. *Anais do XVII Congresso Brasileiro de Automática* (CBA), Juiz de Fora, V. 1, pp. 1–6.

Coelho, L.S.; Silveira, C.C. (2006). Improved Bacterial Foraging Strategy for Controller Optimization Applied to Robotic Manipulator System. *Proceedings of the 2006 IEEE 6^{th} International Symposium on Intelligent Control*, pp. 1276–1281.

Coello C, and Luna E. (2003). Use of particle swarm optimization to design combinational logic Circuits. *Proceedings of the 5th International Conference on Evolvable Systems: from biology to hardware*, ICES 2003. Lecture Notes in Computer Science, Vol. 2606, pp. 398–409.

Cordón, O.; de Viana, I.F.; Herrera, F.; Moreno, L. (2000). A new ACO model integrating evolutionary computation concepts: The best-worst Ant System. *Proceedings of ANTS 2000*, M. Dorigo *et al*. (Eds.), IRIDIA, pp. 22–29.

Correa, E.S.; Freitas, A.A.; Johnson, C.G. (2007). Particle swarm and bayesian networks applied to attribute selection for protein functional classification. *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference*, Companion Material, pp. 2651–2658.

Costa, D.; Hertz, A. (1997). Ants can color graphs. *J. Oper. Res. Soc.*, Vol. 48, pp. 295–305.

Crispin, Y.J. (2009). Cooperative control of multiple swarms of mobile robots with communication constraints. *Lecture Notes in Control and Information Sciences*, Vol. 381, pp. 207–220.

Curkovic, P.; Jerbic, B. (2007). Honey-bees optimization algorithm applied to path planning problem. *International Journal of Simulation Modelling*, Vol. 6, No. 3, pp. 154–164.

Dang, J.; Brabazon, A.; O'Neill, M.; Edelman, D. (2008). Option model calibration using a bacterial foraging optimization algorithm. *Lecture Notes in Computer Science*, Vol. 4974, pp. 113–122.

Dariane, A.B.; Moradi, A.M. (2008). Reservoir Operating by Ant Colony Optimization for Continuous Domains (ACO_R) – Case Study: Dez Reservoir. *Proceedings of World Academy of Science, Engineering and Technology*, Vol. 33, pp. 5–9.

Dawkins, R. (1976). *The Selfish Gene*, Oxford University Press, Oxford.

de Castro, L.N. (2006). Fundamentals of natural computing: basic concepts, algorithms, and applications. CRC Press LLC.

De Jong, K.A. (2006). Evolutionary Computation: A Unified Approach, MIT Press, NewYork.

del Valle, Y.; Venayagamoorthy, G.K.; Mohagheghi, S.; Hernandez, J.-C.; Harley, R.G. (2008). Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 2, pp. 171–195.

Deneubourg, J.-L.; Aron, S; Goss, S.; Pasteels, J.-M. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behav*., Vol. 3, pp. 159–168.

Di Caro, G.; Dorigo, M. (1998). AntNet: distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, Vol. 9, pp. 317–365.

Dorigo, M.; Gambardella, L.M. (1997). Ant colony system: A cooperative learning approach to TSP. *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 53–66.

Dorigo, M.; Maniezzo, V.; Colorni, A. (1991). Positive feedback as a search strategy. Dipartimento di Elettronica, Politecnico di Milano, Italy, *Technical Report TR 91-016*.

Dorigo, M.; Maniezzo, V.; Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics*, Part B, Vol. 26, pp. 29–41.

Dorigo, M.; Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, Vol. 344, No. 2–3, pp. 243–278.

Dréo, J.; Siarry, P. (2002). A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions. *Proceedings of ANTS 2002*, Brussels, Belgium, LNCS, Vol. 2463, pp. 216–221.

Dréo, J.; Siarry, P. (2004). Continuous interacting ant colony algorithm based on dense heterarchy. *Future Generation Computer Systems*, 2004, Vol. 20, pp. 841–856.

Dréo, J.; Siarry, P. (2006). An ant colony algorithm aimed at dynamic continuous optimization. *Appl. Math. Comput*., Vol. 181, pp. 457–467.

Drias, H.; Sadeg, S.; Yahi, S. (2005). Cooperative bees swarm for solving the maximum weighted satisfiability problem, computational intelligence and bioinspired Systems. *Proceedings of the 8^{th} International Workshop on Artificial Neural Networks*.

Eberhart, R.C.; Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the 2000 Congress Evolutionary Computation*, IEEE Press, pp. 84–88

Elbehairy, H.; Elbeltagi, E.; Hegazy, T.; Soudki, K. (2006). Comparison of Two Evolutionary Algorithms for Optimization of Bridge Deck Repairs. *Computer-Aided Civil and Infrastructure Engineering*, Vol. 21, pp. 561–572.

Elbeltagi, E.; Hegazy, T.; Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Journal of Advanced Engineering Informatics*, Elsevier Science, Vol. 19, No. 1, pp. 43–53.

Elbeltagi, E.; Hegazy, T.; Grierson, D. (2007). A modified shuffled-frog-leaping optimization algorithm: Applications to project management, Structure and Infrastructure Engineering. *Structure & Infrastructure Engineering: Maintenance, Management, Life-Cycl*, Vol. 3, No. 1, pp. 53–60(8).

Emara, H.M.; Elshamy, W.; Bahgat, A. (2008). Parameter Identification of Induction Motor Using Modified Particle Swarm Optimization Algorithm. *Proceedings of the IEEE International Symposium on Industrial Electronics*, pp. 841–847.

Eusuff, M.; Lansey, K. (2003). Optimizing of water distribution network design using the shuffled frog-leaping algorithm. *Journal of Water Resources Planning and Manegement*, ASCE, Vol. 129, No. 3, pp. 210–225.

Eusuff, M.; Lansey, K.; Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization*, Vol. 38, No. 2, pp. 129–154.

Falcucci, M.; Junior, J.N.R.S.; Braga, A.P.S.; Nascimento, J.A. (2007). Determinação dos parâmetros ótimos de um controlador PID aplicado na regulação de tensão de geradores síncronos através do método de otimização por enxame de partículas PSO. *Anais do I Simpósio Brasileiro de Inteligência Computacional* (SBIC), Florianópolis.

Fathian, M.; Amiri, B.; Maroosi, A. (2007). Application of honey-bee mating optimization algorithm on clustering. *Applied Mathematics and Computation*, Vol. 190, No. 2, pp. 1502–1513.

Fathian, M.; Amiri, B. (2008). A honeybee-mating approach for cluster analysis. *International Journal of Advanced Manufacturing Technology*, Vol. 38, No. 7–8, pp. 809–821.

Gaing, Z.L. (2004). A Particle swarm optimization approach for optimum design of PID controller in AVR system. *IEEE Transaction on Energy Conversion*, Vol.19, pp. 384–391.

Gambardella, L.M.; Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. *Proceedings of the 12^{th} International Conference on Machine Learning*, A. Prieditis and S. Russell (Eds.), Morgan Kaufmann Publishers, pp. 252–260.

Gambardella, L.M.; Dorigo, M. (1996). Solving symmetric and asymmetric TSPs by ant colonies. *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, T. Baeck *et al*., Eds., IEEE Press, pp. 622–627.

Gambardella, L.M.; Taillard, E.; Dorigo, M. (1999). Ant Colonies for the Quadratic Assignment Problem. *Journal of the Operational Research Society*, Vol. 50, pp. 167–176.

Gambardella, L.M. & Dorigo, M. (2000). Ant colony system hybridized with a new local search for sequential ordering problem. *INFORMS Journal on Computing*, Vol. 12, No. 3, pp. 237–255.

Gazi, V.; Passino, K.M. (2003). Stability analysis of swarms. *IEEE Transactions on Automatic Control*, Vol. 48, No. 4, pp. 692–697.

Gazi, V.; Passino, K.M. (2004). Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, Vol. 34, No. 1, pp. 539–557.

Geis, M.; Middendorf, M. (2007). An ant colony optimizer for melody creation with baroque harmony. *Proceedings of the 2007 IEEE Congress on Evolutionary Computation* (CEC 2007), pp. 461–468.

Goldberg, D.E. (1989). *Generic Algorithm in search. optimization and machine learning*, Addison-Wesley.

Goldberg, D.E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers, Norwell, MA, USA.

Grosan, C.; Abraham, A.; Han, S.; Gehlbukh, A. (2005). Hybrid particle swarm – evolutionary algorithm for search and optimization. *Proceedings of the 4^{th} Mexican International Conference on Artificial Intelligence*, Lecture Notes in Computer Science, pp. 623–632.

Guney, K.; Basbug, S. (2008a). Interference suppression of linear antenna arrays by amplitude-only control using a bacterial foraging algorithm. *Progress in Electromagnetics Research*, Vol. 79, pp. 475–497.

Guney, K.; Basbug, S. (2008b). Phase-only pattern nulling of linear antenna arrays with the use of a bacterial foraging algorithm. *Neural Network World*, Vol. 18, No. 4, pp. 257–273.

Guo Q.-J., Yu H.-B., & Xu A.-D. (2006), A Hybrid PSO-GD based Intelligent Method for Machine diagnosis, Digital Signal Processing, Vol. 16, No. 4, pp. 402–18.

Gutjahr, W.J. (2000). A graph based ant-system and its convergence. *Future Generation Computer Systems*, Vol. 16, pp. 873–888.

Gutjahr, W.J. (2002). ACO algorithms with guaranteed convergence to the optimal solution. *Information Processing Letters*, Vol. 82, No. 3, pp. 145–153.

Hani, Y.; Amodeo, L.; Yalaoui, F.; Chen, H. (2007). Ant colony optimization for solving an industrial layout problem. *EJOR European Journal of Operational Research*, Vol. 183, Issue 2, pp. 633–642.

Hu, X.M.; Zhang, J.; Li, Y. (2008). Orthogonal methods based ant colony search for solving continuous optimization problems. *Journal of Computer Science and Technology*, Vol. 23, No.1, pp. 2–18.

Izquierdo, J.; Montalvo, I.; Peìrez, R.; Fuertes, V.S. (2009). Forecasting pedestrian evacuation times by using swarm intelligence. *Physica A: Statistical Mechanics and its Applications*, Vol. 388, No. 7, pp. 1213–1220.

Huynh, T.-H. (2008). A modified shuffled frog leaping algorithm for optimal tuning of multivariable PID controllers. *Proceedings of the IEEE International Conference on Industrial Technology*, paper no. 4608439.

Jalalinejad, F.; Jalali-Farahania, F.; Mostoufi, N.; Sotudeh-Gharebagh, R. (2007). Ant Colony Optimization: A Leading Algorithm in Future Optimization of Chemical Processes. *Proceedings of the 17th European Symposium on Computer Aided Process Engineering* (ESCAPE17).

Juang, C.-F.; Wang, C.-Y.; (2009). A self-generating fuzzy system with ant and particle swarm cooperative optimization. *Expert Systems with Applications*, Vol. 36 (3 PART 1), pp. 5362–5370.

Kara A. (2004). Imitation of Bee Reproduction as a Crossover Operator in Genetic Algorithms, PRICAI 2004, Lecture Notes in Artificial Intelligence, Vol. 3157, C. Zhang, H.W. Guesgen, W.K. Yeap (Eds.), Springer Verlag, pp. 1015–1016.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical Report TR06*, Erciyes University.

Karaboga, D.; Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, Vol. 39, No. 3, pp. 459–471.

Karaboga, N. (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. *Journal of the Franklin Institute*, Article in Press.

Kashan, A.H.; Karimi, B. (2009). A discrete particle swarm optimization algorithm for scheduling parallel machines. *Computers and Industrial Engineering*, Vol. 56, No. 1, pp. 216–223.

Kennedy, J.; Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948.

Kennedy J. & Eberhart R. (1997), A Discrete Binary Version of the Particle Swarm Algorithm. *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 5, pp. 4104–4108.

Kennedy, J.; Eberhart, R.C.; Shi., Y. (2001) *Swarm Intelligence*, San Francisco: Morgan Kaufmann/ Academic Press.

Kim, D.H.; Cho, J.H. (2005). Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization. *Advances in Web Intelligence*. In: *Lecture Notes in Computer Science*, Vol. 3528, Springer Verlag, pp. 231–235.

Kim, D.H.; Abraham, A.; Cho, J.H. (2007). A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Information Sciences*, Vol. 177, No. 18, pp. 3918–3937.

Kong, M.; Tian, P. (2005). A binary ant colony optimization for the unconstrained function optimization problem. *Proceedings of the International Conference on Computational Intelligence and Security* (CIS'05), Xi'an, China, LNAI, Vol. 3801, pp.682–687.

Kong, M.; Tian, P. (2006). A direct application of ant colony optimization to function optimization problem in continuous domain. *Proceedings of ANTS 2006*, Brussels, Belgium, LNCS, Vol. 4150, pp. 324–331.

Korani, W. (2008). Bacterial Foraging Oriented by Particle Swarm Optimization Strategy for PID Tuning. *Proceedings of GECCO 2008: Genetic and Evolutionary Computation Conference*, Atlanta, Georgia, pp. 1823–1826.

Korte, B.H.; Vygen, J. (2008). *Combinatorial optimization theory and algorithms*. Springer, 4th edition.

Koudil, M.; Benatchba, K.; Tarabet, A.; Sahraoui, E.B. (2007). Using Artificial Bees to Solve Partitioning and Scheduling Problems in Codesign. *Applied Mathematics and Computation*, Vol. 186, No. 2, pp. 1710–1722.

Koza, J.R.; Keane, M.A.; Streeter, M.J.; Mydlowec, W. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Springer Science+Business Media, New York.

Kwok, N.M.; Ha, Q.P.; Liu, D.; Fang, G. (2009). Contrast enhancement and intensity preservation for gray-level images using multiobjective particle swarm optimization. *IEEE Transactions on Automation Science and Engineering*, Vol. 6, No. 1, pp. 145–155.

Lemmens, N.; De Jong, S.; Tuyls, K.; Noweì, A. (2008). Bee behaviour in multi-agent systems (a bee foraging algorithm*). Lecture Notes in Artificial Intelligence*, Vol. 4865, pp. 145–156.

Leiviskä, K.; Joensuu, I. (2006). Chemotaxis for controller tuning. *Proceedings of the 2nd Annual Symposium of the Nature-inspired Smart Information Systems* (NiSIS).

Lenin, K.; Mohan, M.R. (2006). Ant Colony Search Algorithm for Optimal Reactive Power Optimization. *Serbian Journal of Electrical Engineering*, Vol. 3, No. 1, pp. 77–88.

Lessing, L.; Dumitrescu, I.; Stutzle, T. (2004). A Comparison between ACO Algorithms for the Set Covering Problem. *Proceedings of the ANTS' 2004*, Brussels, Belgium, LNCS, Vol. 3172, pp. 1–12.

Li, K.-S.; Pan, W.-F.; Tang, M.-D.; Wang, F. (2008). Evolutionary algorithm for solving complex problem based on queen-bee mating. *Xitong Fangzhen Xuebao / Journal of System Simulation*, Vol. 20, No. 7, pp. 1707–1712+1757.

Liao C.-J., Tseng C.-T. & Luarn P. (2007). A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems. *Computer & Operations Research*, Vol. 34, pp. 3099–3111.

Liong, S.Y.; Atiquzzaman, Md. (2004). Optimal Design of Water Distribution Network using Shuffled Complex Evolution. *Journal of The Institution of Engineers*, Singapore, Vol. 44, No. 1, pp. 93–107.

Liu, Y.; Abraham, A. (2005). Fuzzy adaptive turbulent particle swarm optimization. *Proceedings of the 5th International Conference on Hybrid Intelligent Systems*, pp. 6–9.

Liu, Y.; Passino, K.M. (2002). Biomimicry of social foraging bacteria for distributed optimization: models, principles and emergent behaviors. *Journal of Optimization Theories and Applications*, Vol. 115, No. 3, pp. 603–628.

Lopes, H.S.; Dalle Molle, V.L.; Lima, C.R.E. (2007). Aplicação da otimização por colônia de formigas ao problema de roteamento de múltiplos veículos de capacidade limitada. *Anais do Simpósio Brasileiro de Inteligência Computacional* (SBIC), Florianópolis.

Lovbjerg, M.; Rasmussen, T.K.; Krink, T. (2001). Hybrid Particle Swarm Optimizer with Breeding and Subpopulations. *Proceedings of the third Genetic and Evolutionary Computation Conference*, pp. 469–476.

Low, K.S.; Nguyen, H.A.; Guo, H. (2008).A particle swarm optimization approach for the localization of a wireless sensor network. *IEEE International Symposium on Industrial Electronics*, art. no. 4677205, pp. 1820–1825.

Lu, X.; Zhou, Y. (2008). A novel global convergence algorithm: Bee collecting pollen algorithm. *Lecture Notes in Artificial Intelligence*, Vol. 5227, pp. 518–525.

Lu, Z.S.; Hou, Z.R. (2004). Particle Swarm Optimization with Adaptive Mutation. *Acta Electronica Sinica*, Vol. 32, No. 3, pp. 416–420.

Luenberger, D.G. (2003). *Linear and Nonlinear Programming*. Springer, 2nd. edition.

Lucic, P.; Teodorovic, D. (2001). Bee System: Modeling Combinatorial Optimization Transportation Engineering Problems by Swarm Intelligence. *Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis*, pp. 441–445.

Lucic, P.; Teodorovic, D. (2003). Computing with bees: attacking complex transportation engineering problems. *International Journal on Artificial Intelligence Tools*, Vol. 12, No. 3, pp. 375–394.

Luna, E.H.; Coello, C.A.C.; Aguirre, A.H. (2004). On the use of a population-based particle swarm optimizer to design combinational logic circuits. *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, pp. 183–190.

Maitra, M.; Chatterjee, A. (2008). A novel technique for multilevel optimal magnetic resonance brain image thresholding using bacterial foraging. *Journal of the International Measurement Confederation*, Vol. 41, No. 10, pp. 1124–1134.

Majhi, B.; Panda, G. (2007). Bacterial Foraging based Identification of Nonlinear Dynamic System. *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 1636–1641.

Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, Vol. 11, No. 4, pp. 358–369.

Marinakis, Y.; Marinaki, M.; Dounias, G. (2008a). Honey bees mating optimization algorithm for the vehicle routing problem. *Studies in Computational Intelligence*, Vol. 129, pp. 139–148.

Marinakis, Y.; Marinaki, M.; Matsatsinis, N. (2008b). Honey Bees Mating Optimization for the Location Routing Problem. *Proceedings of the 2008 IEEE International Engineering Management Conference, Europe* (IEMC-Europe 2008): *Managing Engineering, Technology and Innovation for Growth*, paper no. 4618013.

Martens, D.; Huysmans, J.; Setiono, R.; Vanthienen, J.; Baesens, B. (2008). Rule extraction from support vector machines: An overview of issues and application in credit scoring. *Studies in Computational Intelligence*, Vol. 80, pp. 33–63.

Mathur, M.; Karale, S.B.; Priye, S.; Jyaraman, V.K.; Kulkarni, B.D. (2000). Ant colony approach to continuous function optimization. *Ind. Eng. Chem. Res.*, Vol. 39, pp. 3814–3822.

Merkle, D.; Middendorf, M. (2003). Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Applied Intelligence*, Vol. 18, No. 1, pp. 105–111.

Millonas, M. M. (1994). Swarms, phase transitions, and collective intelligence. In C.G. Langton (Ed.), *Artificial Life III*, pp. 417–445. Reading, MA: Addison-Wesley.

Miranda, V.; Fonseca, N. (2002). EPSO – evolutionary particle swarm optimization, a new algorithm with applications in power systems. *Proceedings of the Asia Pacific Transmission and Distribution Conference and Exhibition*, Vol. 2, pp. 745–750.

Mishra, S. (2007). Bacteria foraging based solution to optimize both real power loss and voltage stability limit. *Proceedings of the 2007 IEEE Power Engineering Society General Meeting*, PES, paper no. 4275250.

Mishra, S.; Reddy, G.D.; Rao, P.E.; Santosh, K. (2007). Implementation of new evolutionary techniques for transmission loss reduction. *Proceedings of the IEEE Congress on Evolutionary Computation* (CEC 2007), paper no. 4424762, pp. 2331–2336.

Mo N., Zou Z. Y., Chan K. W., & Pong T. Y. G. (2007), Transient stability constrained optimal power flow using particle swarm optimization. *IET Generation, Transmission, and Distribution*, Vol. 1, No. 3, pp. 476–483.

Monson, C.K.; Seppi, K.D. (2004). The Kalman swarm: A new approach to particle motion in swarm optimization. *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 140–150.

Muller, S.D.; Marchetto, J.; Airaghi, S.; Koumoutsakos, P. (2002). Optimization based on bacterial chemotaxis. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 1, pp. 16–29.

Müller, V.; Furtado, J.C.; Neis, J.F.; Crossetti, G.L. (2006). Otimização do problema de layout de facilidades através da técnica enxame de partículas utilizando a modelagem attractor-repeller. *Anais do XXVI Encontro Nacional de Engenharia de Produção* (ENEGEP), Fortaleza.

Nakrani, S.; Tovey, C. (2003). On Honey Bees and Dynamic Allocation in an Internet Server Colony. *Proceedings of* 2^{nd} *International Workshop on The Mathematics and Algorithms of Social Insects*, Atlanta, Georgia, USA.

Nezamabadipour, H.; Saryazdi, S.; Rashedi, E. (2006). Edge detection using ant algorithms. *Soft Computing*, Vol. 10, pp. 623–628.

Niu, B.; Zhu, Y.; He, X.;Zeng, X. (2006). Optimum Design of PID Controllers Using Only a Germ of Intelligence. *Proceedings of the* 6^{th} *World Congress on Intelligent Control and Automation*, pp. 3584–3588.

Parpinelli, R.S.; Lopes, H.S.; Freitas, A.A. (2002). Data Mining with an Ant Colony Optimization Algorithm. *IEEE Trans on Evolutionary Computation, special issue on Ant Colony Algorithms*, Vol. 6, No. 4, pp. 321–332.

Passino, K.M. (2000). Distributed optimization and control using only a germ of intelligence. *Proceedings of the 2000 IEEE International Symposium on Intelligent Control*, pp. 5–13.

Passino, K.M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, Vol. 22, No. 3, pp. 52–67.

Pei, Z.; Hua, X.; Han, J. (2008). The clustering algorithm based on particle swarm optimization algorithm. *Proceedings of the International Conference on Intelligent Computation Technology and Automation* (ICICTA 2008), art. no. 4659461, pp. 148–151.

Pelikan, M.; Sastry, K.; Cantú-Paz, E. (Eds.) (2006). Scalable Optimization via Probabilistic Modeling. *Studies in Computational Intelligence*, Vol. 33, Springer.

Pham, D.T.; Kog, E.; Ghanbarzadeh, A.; Otri, S.; Rahim, S.; Zaidi, M. (2006a). The Bees Algorithm – A Novel Tool for Complex Optimisation Problems. *Proceedings of 2nd International Virtual Conference on Intelligent Production Machines and Systems* (IPROMS 2006), Oxford, Elsevier.

Pham, D.T.; Otri, S.; Ghanbarzadeh, A.; Koç, E. (2006b). Application of the Bees Algorithm to the Training of Learning Vector Quantisation Networks for Control Chart Pattern Recognition. *ICTTA'06 Information and Communication Technologies*, pp. 1624–1629.

Pham, D.T.; Koç, E.; Ghanbarzadeh, A.; Otri, S. (2006c). Optimisation of the Weights of Multi-Layered Perceptions Using the Bees Algorithm. *Proceedings of 5th International Symposium on Intelligent Manufacturing Systems*, Sakarya, Turkey, pp. 38–46.

Pham, D.T.; Soroka, A.J.; Ghanbarzadeh, A.; Koç, E.; Otri, S.; Packianather M. (2006d). Optimising Neural Networks for Identification of Wood Defects Using the Bees Algorithm. *IEEE International Conference on Industrial Informatics*, Vol. 8, pp. 1346–1351.

Pham, D.T.; Afify, A.; Koç, E. (2007a). Manufacturing cell formation using the bees algorithm. *Proceedings of the Innovative Production Machines and Systems Virtual Conference*.

Pham, D.T.; Darwish, A.H..; Eldukhr, E.; Otri, S. (2007b). Using the bees algorithm to tune a fuzzy logic controller for a robot gymnasta. *Proceedings of the Innovative Production Machines and Systems Virtual Conference*.

Pham, D.T.; Ghanbarzadeh, A.; Koç, E.; Otri, S.; Rahim, S.; Zaidi, M. (2007c). The bees algorithm – the novel tool for complex optimisation problems. *Proceedings of the Innovative Production Machines and Systems Virtual Conference*.

Pham, D.T.; Ghanbarzadeh, A. (2007d). Multi-objective optimisation using the bees algorithm. *Proceedings of the Innovative Production Machines and Systems Virtual Conference*.

Pham, D.T.; Muhamad, Z.; Mahmuddin, M.; Ghanbarzadeh, A.; Koç, E.; Otri, S. (2007e). Using the bees algorithm to optimize a support vector machine for wood defect classification. *Proceedings of the Innovative Production Machines and Systems Virtual Conference*.

Pham, D.T.; Castellani, M.; Fahmy, A.A. (2008). Learning the inverse kinematics of a robot manipulator using the Bees Algorithm. *Proceedings of the IEEE International Conference on Industrial Informatics* (INDIN), paper no. 4618151, pp. 493–498.

Poli, R.; Kennedy, J.; Blackwell, T. (2007). Particle Swarm Optimisation: an overview. *Swarm Intelligence Journal*, Vol. 1, No. 1, pp. 33–57.

Pourtakdoust, S.H.; Nobahari, H. (2004). An extension of ant colony system to continuous optimization problems. *Proceedings of ANTS 2004*, Brussels, Belgium, LNCS, Vol. 3172, pp. 294–301.

Quijano N.; Passino K.M. (2007). Honey Bee Social Foraging Algorithms for Resource Allocation Theory and Application. *American Control Conference*, New York City, USA.

Qin, L.D.; Jiang, Q.Y.; Zou, Z.Y.; Cao, Y.J. (2004). A Queen-Bee Evolution Based on Genetic Algorithm for Economic Power Dispatch. *UPEC 2004 39th International Universities Power Engineering Conference*, pp. 453–456.

Rahimi-Vahed, A.; Mirzaei, A.H. (2007). Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm. *Soft Computing*, Vol. 12, No. 5, pp. 435–452.

Ru, N.; Jianhua, Y. (2008). A GA and Particle Swarm Optimization based hybrid algorithm. *Proceedings of the 2008 IEEE Congress on Evolutionary Computation* (CEC 2008), art. no. 4630925, pp. 1047–1050.

Saber, A.Y.; Venayagamoorthy, G.K. (2008). Economic load dispatch using bacterial foraging technique with particle swarm optimization biased evolution. *2008 IEEE Swarm Intelligence Symposium*, art. no. 4668291.

Samanta, B.; Nataraj, C. (2009). Use of particle swarm optimization for machinery fault detection. *Engineering Applications of Artificial Intelligence*, Vol. 22, No. 2, pp. 308–316.

Schoofs, L.; Naudts, B. (2000). Ant colonies are good at solving constraint satisfaction problems. *Proceedings of the 2000 Congress on Evolutionary Computation*, San Diego, CA, pp. 1190–1195.

Schoonderwoerd, R.; Holland, O.; Bruten, J. & Rothkrantz, L. (1996). Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, Vol. 5, No. 2, pp. 169–207.

Serapião, A.B.S. (2009). PID Tuning By Swarm Optimization Strategies. *Proceedings of the 8th Brazilian Conference on Dynamics, Control and their Applications* (DINCON 2009), Bauru, São Paulo.

Shi, Y.; Eberhart, R.C. (1998). A Modified Particle Swarm Optimizer. In: *IEEE World Congress on Computational Intelligence*, Anchorage, Alaska, pp.69–73.

Silva, R.C.; Fonseca, A.M.; Delgado, M.R.B.S. (2007). Otimização por Colônia de Formigas para o Roteamento e Consumo de Energia em Redes de Sensores sem Fio. *Anais do I Simpósio Brasileiro de Inteligência Computacional* (SBIC), Florianópolis.

Socha K. (2004). ACO for continuous and mixed-variable optimization. *Proceedings of ANTS 2004*, Brussels, Belgium, LNCS, Vol. 3172, pp. 25–36.

Socha, K.; Dorigo, M. (2008). Ant colony optimization for continuous domains. *Eur. J. Oper. Res.*, Vol. 185, No. 3, pp. 1155–1173.

Solimanpur, M.; Vrat, P.; Shankar, R. (2004). Ant Colony optimization algorithm to the intercell layout problem in cellular manufacturing. *European Journal of Operational Research*, Vol. 157, pp. 592–606.

Sousa, T.; Silva, A.; Neves, A. (2004). Particle Swarm based Data Mining Algorithms for classification tasks. *Parallel Computing*, Vol. 30, pp. 767–783.

Stutzle, T.; Hoos, H.H. (1996). Improving the Ant System: A detailed report on the MAX–MIN Ant System. FG Intellektik, FB Informatik, TU Darmstadt, Germany, *Technical Report AIDA–96–12*.

Stutzle, T.; Dorigo, M. (2002). A Short Convergence Proof for a Class of ACO Algorithms, IEEE *Transactions on Evolutionary Computation*, Vol. 6, No. 4, pp. 358–365.

Sun, X.; Wang, Z.; Zhang, D. (2008). A Web document classification method based on shuffled frog leaping algorithm. *Proceedings of the 2nd International Conference on Genetic and Evolutionary Computing, WGEC 2008*, paper no. 4637428, pp. 205–208.

Sung H.J. (2003). Queen-Bee Evolution for Genetic Algorithms. *Electronic Letters*, Vol. 39, No. 6, pp. 575–576.

Tan, G.; Zeng, Q.; He, S.; Cai, G. (2005). Adaptive and Robust Design for PID Controller Based on Ant System Algorithm. *Lecture Notes in Computer Science*, Vol. 3612, pp. 915–924.

Tang, W.J.; Wu, Q.H.; Saunders, J.R. (2007). A Bacterial Swarming Algorithm for Global Optimization. *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 1207–1212.

Tang, W.J.; Li, M.S.; Wu, Q.H.; Saunders, J.R. (2008). Bacterial foraging algorithm for optimal power flow in dynamic environments. *IEEE Transactions on Circuits and Systems I*: Regular Papers, Vol. 55, No. 8, pp. 2433–2442.

Tavares Neto, R.F.; Coelho, L.S. (2003). Proposta de Planejamento de Rotas para Robôs de Inspeção Usando um Algoritmo de Colônia de Formigas. *Anais do VI Simpósio Brasileiro de Automação Inteligente* (SBAI), Bauru, SP.

Tavares Neto, R.F.; Coelho, L.S. (2004). Planejamento de Rotas de Entrega Usando uma Abordagem de Algoritmo de Colonia de Formigas com Lista Tabu Suprimida. *Anais do XV Congresso Brasileiro de Automática* (CBA), Gramado, RS.

Tavares Neto, R.F.; Coelho, L.S. (2005). Planejamento de Rotas para Robos de Inspecao Usando um Algoritmo hibrido de Colonia de Formigas e Algoritmo Cultural. *Anais do VII Simpósio Brasileiro de Automação Inteligente* (SBAI).

Tebaldi, A.; Coelho, L.S.; Lopes Junior, V. (2006). Detecção de falhas em estruturas inteligentes usando otimização por nuvem de partículas: fundamentos e estudo de casos. *Sba Controle & Automação*, Vol. 17, No. 3, pp. 312–330.

Teodorovic, D.; Dell'Orco, M. (2005). Bee colony optimization – a cooperative learning approach to complex transportation problems. *Proceedings of the 10th EWGT Meeting and 16th Mini-EURO Conference*.

Thanh, B.T.; Parnichkun, M. (2008). Balancing control of bicyrobo by particle swarm optimization-based structure-specified mixed H2/H∞ control. *International Journal of Advanced Robotic Systems*, Vol. 5, No. 4, pp. 395–402.

Tillet, J.; Rao, T.M.; Sahin, F.; Rao, R. (2005). Darwian particle swarm optimization. *Proceedings of the 2nd Indian International Conference on Artificial Intelligence*, pp. 1474–1487.

Venayagamoorthy, G.K.; Smith, S.C.; Singhal, G. (2007), Particle swarm-based optimal partitioning algorithm for combinational CMOS circuits. *Engineering Applications of Artificial Intelligence*, Vol. 20, pp. 177–184.

Vieira, S.M.; Sousa, J.M.C.; Runkler, T.A. (2007). Ant colony optimization applied to feature selection in fuzzy classifiers. *Lecture Notes in Artificial Intelligence*, Vol. 4529, pp. 778–788.

Wang, J.; Che, J.; Sun, D.; Liang, J. (2008). ARMA Model identification using Particle Swarm Optimization Algorithm. *Proceedings of the International Conference on Computer Science and Information Technology*, ICCSIT 2008, art. no. 4624865, pp. 223–227.

Wang, L.; Singh, C. (2009). Reserve-constrained multiarea environmental/economic dispatch based on particle swarm optimization with local search. *Engineering Applications of Artificial Intelligence*, Vol. 22, No. 2, pp. 298–307.

Watkins, C. (1989). *Learning from Delayed Rewards*, Thesis, University of Cambidge, England.

Wedde H.F.; Farooq M.; Zhang Y. (2004). BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior, Ant Colony, Optimization and Swarm Intelligence. In: *Proceedings of the 4th International Workshop*, M. Dorigo (Ed.), Lecture Notes in Computer Science, Vol. 3172, pp. 83–94.

Wedde, H.F.; Lehnhoff, S.; Van Bonn, B.; *et al*. (2007). A novel class of multi-agent algorithms for highly dynamic transport planning inspired by honey bee behavior. *Proceedings of the IEEE Symposium on Emerging Technologies and Factory Automation* (ETFA), paper no. 4416912, pp. 1157–1164.

Wickramasinghe, W.R.M.U.K.; Li, X. (2008). Choosing leaders for multi-objective PSO algorithms using differential evolution. *Lecture Notes in Artificial Intelligence*, Vol. 5361, pp. 249–258.

Wodrich, M.; Bilchev, G. (1997). Cooperative distributed search: The ant's way. *Control and Cybernetics*, Vol. 3, pp. 413–446.

Wolsey, L.A. (1998). *Integer Programming*. Wiley.

Wong, L-P.; Low, M.Y.H.; Chong, C.S. (2008). A Bee Colony Optimization Algorithm for Traveling Salesman Problem. *Proceedings of the* 2^{nd} *Asia International Conference on Modelling & Simulation*, pp. 818–823.

Wysota, M.; Jagodziska, K.; Walkowiak, M. (2008). Sidelob suppression in unequally spaced antenna arrays. *Proceedings of the 2008 1st International Conference on Information Technology*, IT 2008, art. no. 4621695.

Xu, C.; Zhang, Q.; Li, J.; Zhao, X. (2008). A bee swarm genetic algorithm for the optimization of DNA encoding. *Proceedings of the 3rd International Conference on Innovative Computing Information and Control* (ICICIC'08), paper no. 4603224.

Xu, G.; Gao, J. (2008). Weak signal detection based on a new Matching Pursuit method. *Proceedings of the 7th International Conference on Machine Learning and Cybernetics*, ICMLC 2, art. no. 4620557, pp. 1036–1040.

Yang, C.; Chen, J.; Tu, X. (2007a). Algorithm of marriage in honey bees optimization based on the local particle swarm optimization. *Journal of Information and Computational Science*, Vol.4, No. 3, pp. 961–973.

Yang, C.; Tu, X.; Chen, J. (2007b). Algorithm of marriage in honey bees optimization based on the wolf pack search. *Proceedings of the 2007 International Conference on Intelligent Pervasive Computing* (IPC 2007), paper no. 4438476, pp. 462–467.

Yang, C.; Chen, J.; Tu, X. (2007c). Algorithm of fast marriage in honey bees optimization and convergence analysis. *Proceedings of the IEEE International Conference on Automation and Logistics* (ICAL 2007), paper no. 4338865, pp. 1794–1799.

Yang X.S. (2005). Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. IWINAC 2005, *Lecture Notes in Computer Science*, Vol. 3562, Yang, J. M. and J.R. Alvarez (Eds.), Springer-Verlag, Berlin Heidelberg, pp. 317–323.

Yijian, L.; Yanjun, F. (2008). Optimization design of PID controller parameters based on improved E.Coli foraging optimization algorithm. *Proceedings of the IEEE International Conference on Automation and Logistics* (ICAL 2008), paper no. 4636151, pp. 227–231.

Ying, K.-C.; Liao, C.-J. (2004). An ant colony system for permutation flow-shop sequencing, *Computers & Operations Research*, Vol. 31, pp. 791–801.

Zhang, W.; Xie, X. (2003). DPSO: Hybrid particle swarm with differential evolution operator. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 3816–3821.

Zhang, X.; Hu, X.; Cui, G.; Wang, Y.; Niu, Y. (2008). An improved shuffled frog leaping algorithm with cognitive behavior. *Proceedings of the World Congress on Intelligent Control and Automation* (WCICA), paper no. 4592798, pp. 6197–6202.

Zhao, L.; Yang, Y. (2009). PSO-based single multiplicative neuron model for time series prediction. *Expert Systems with Applications*, Vol. 36 (2 PART 2), pp. 2805–2812.