IEEE.org | Biblioteca Digital IEEE *Xplore* | IEEE-SA | Espectro IEEE | Mais sites

Carrinho (0) | Criar Conta em portugues-Brasil | Login pessoal

baixar PDF

Acesso fornecido por:
UNIVERSIDADE FEDERAL DO PIAUI
Sair

Squeaky toy **Minhas configurações** **Obter ajuda**

Um algoritmo de otimização de colônia de formigas aprimorado com algoritmo genético incorporado para o problema do vendedor ambulante

4 Autor (es) Fanggeng Zhao ; Jinyan Dong ; Sujian Li ; Jiangsheng Sun **Visualizar todos os autores**

1 Papel Citação

215 Cheio Exibições de texto

Exportar para Collabratec

Alertas

Gerenciar alertas de conteúdo

Adicionar aos Alertas de Citação

Abstrato

Seções do documento

EU. Introdução

II. ACO PARA O PROBLEMA DE VENDEDOR DE VIAGEM

III O algoritmo proposto para o TSP

IV. Resultados e análises computacionais

V. Conclusão

Autores

Figuras

Referências

Citações

Palavras-chave

Métricas

Mais como isso

Resumo: Neste trabalho propomos um algoritmo de otimização de colônia de formigas aprimorado com algoritmo genético incorporado para resolver o problema do vendedor ambulante. A idéia principal é deixar ... **Ver o seguimento**

Metadados

Abstrato:

Neste trabalho, propusemos um algoritmo de otimização de colônia de formigas aprimorado com algoritmo genético incorporado para resolver o problema do vendedor ambulante. A ideia principal é deixar o algoritmo genético simular o mecanismo de consulta, que pode ter mais chances de encontrar uma solução melhor, para otimizar as soluções encontradas pelas formigas. No algoritmo proposto, empregamos uma nova maneira de construir uma solução e projetamos um operador de crossover melhorado para consulta no algoritmo genético embutido. Resultados experimentais mostraram que o algoritmo proposto poderia encontrar melhores soluções de instâncias de referência dentro de menos iterações do que os algoritmos de colônia de formigas existentes.

Publicado em: 2008 7º Congresso Mundial de Controle Inteligente e Automação

Data da Conferência: 25 a 27 de junho de 2008

Data adicionada ao IEEE *Xplore* : 08 de agosto de 2008

Informação ISBN:

Número de Acesso INSPEC **:** 10344009

DOI: 10.1109 / WCICA.2008.4594163

Editora: IEEE

Local da Conferência: Chongqing, China

Conteúdo

SEÇÃO I. Introdução

O algoritmo de otimização de colônia de formigas (ACO) é um sistema multiagente no qual o comportamento de cada formiga é inspirado no comportamento de forrageamento de formigas reais para resolver problemas de otimização. A idéia principal da ACO é usar o equivalente

Mais como isso

Um Algoritmo Genético Equilibrando Exploração e Exploração para o Problema do Vendedor Viajante
2008 Quarta Conferência Internacional sobre Computação Natural
Publicado em: 2008

Uma Nova Abordagem para Resolver o Problema do Vendedor Ambulante Usando Algoritmo Genético Baseado no Cruzamento Heurístico e Operador de Mutação
Conferência Internacional de Soft Computing e Reconhecimento de Padrões 2009
Publicado: 2009

Veja mais

Veja as principais organizações de patentes em tecnologias mencionadas neste artigo

Clique para expandir

Provided by: InnovationQ PLUS POWERED BY IEEE AND IP.COM A PATENT SEARCH AND ANALYTICS TOOL

baixar PDF

da trilha de feromônio usada por formigas reais como um meio de cooperação e comunicação entre uma colônia de formigas artificiais. O algoritmo ACO tem sido aplicado com sucesso a vários problemas de otimização combinatória NP-hard, como o problema do caixeiro viajante (TSP) [9] , o problema de atribuição quadrática (QAP) [14] , o problema de roteamento de veículos (VRP) [13] e o problema de agendamento de job-shop (JSP) [3] , desde que foi introduzido por Dorigo [9]. Entre esses problemas, o TSP desempenha um papel importante no algoritmo ACO porque quase todas as pesquisas sobre o ACO foram testadas neste problema. A TSP foi escolhida principalmente por três razões [5] : (I) é um problema ao qual o ACO é facilmente adaptável, (II) é um dos problemas NP-hard mais estudados na otimização combinatória, e (III) é compreensão mais fácil. Assim, nós aqui focamos no TSP como domínios de aplicação para o algoritmo proposto.

Como o primeiro algoritmo ACO, chamado Ant System, não era competitivo com algoritmos de última geração para o TSP, muitas variantes, incluindo $Um\ s_{e\,l\,i\,t\,e}Um, SMMAS$ e cAS et al, do sistema Ant foram propostos por pesquisadores. E até agora, o algoritmo ACO ainda é confrontado com a forma de melhorar seu desempenho.

Neste estudo, apresentamos uma otimização de colônia de formigas melhorada com algoritmo genético embutido (GA) para o TSP para melhorar o desempenho do algoritmo ACO. Neste algoritmo, o algoritmo genético embutido é usado para simular o mecanismo de consulta entre formigas, a fim de otimizar as soluções encontradas por eles, e então as trilhas de feromônio são atualizadas. A adoção do GA integrado não apenas acelera a pesquisa, mas melhora a qualidade da solução.

O restante deste artigo está organizado da seguinte forma. A seção 2 fornece uma breve revisão dos algoritmos da ACO que foram propostos para o TSP. As seções 3 e 4 apresentam o algoritmo proposto e seus resultados computacionais, respectivamente. Finalmente, a Seção 5 tira conclusões deste estudo.

SEÇÃO II
ACO PARA O PROBLEMA DE VENDEDOR DE VIAGEM

O problema do vendedor ambulante é a primeira aplicação do algoritmo ACO. O TSP geral pode ser representado por um gráfico completo $G = (N, A)$ com N sendo o conjunto de cidades, e A sendo o conjunto de arcos conectando totalmente os nós. Cada arco $(i, j) \in A$ é atribuído um valor d_{eu} que representa a distância entre as cidades Eu e j. O TSP é então o problema de encontrar um tour fechado mais curto visitando cada um dos $n = |N|$ nós de G exatamente uma vez.

Em 1991, Dorigo et al [9] propuseram o primeiro algoritmo ACO denominado Ant System (AS). Em AS, as formigas artificiais constroem soluções (tours) do TSP movendo-se de uma cidade para outra. O algoritmo é executado $t_{\text{máximo}}$ iterações, no seguinte, indexadas por t. Durante cada iteração, m formigas constroem um tour em execução n etapas nas quais uma regra de decisão probabilística (transição de estado) é aplicada. Na prática, quando no nó Eu a formiga escolhe o nó j para mover e o arco (i, j) é adicionado ao tour em construção. Esta etapa é repetida até que a formiga tenha completado sua turnê. De acordo com a maneira como as trilhas de feromônio são atualizadas, o algoritmo AS pode ser dividido em três tipos: *densidade de formigas, quantidade de formigas e ciclo de formigas* [2] , [4] , [9] , [10] . Na densidade de formigas e na formiga-quantidade, as formigas depositam o feromônio enquanto constroem uma solução, enquanto que as formigas de ciclo-formigas depositam o feromônio depois de construírem uma excursão completa. Numerosos experimentos são executados em instâncias de benchmark [4] , [9] , [10]indicam que o desempenho do ant-cycle é

muito melhor que o dos outros dois algoritmos. Embora o desempenho

do AS seja limitado em comparação com outras heurísticas, especialmente em grandes problemas do TSP, ele é o protótipo de vários algoritmos que encontraram muitos aplicativos interessantes e bem-sucedidos.

baixar PDF

Um set é a primeira melhora de AS [1] , [7] , [9] , em que a melhor formiga tem mais “peso” em contribuir para as trilhas de feromônio do que outras formigas. Bullnheimer et al [1] propuseram uma versão baseada em rank do AS, denominada*Um rank*. Diferente de*Um tudo* tanto a melhor formiga como outras formigas que fornecem boas soluções podem adicionar trilhas de feromônio em *Um rank* e, da mesma forma, as formigas “melhores” têm o maior “peso”. Os resultados computacionais das instâncias de benchmark mostram as efetividades dessas medidas.

A fim de evitar a estagnação em que todas as formigas estão presas dentro de um ótimo local, Stlitzle e Hoos [18] , [19] , [20] apresentaram o AS *Max-Max* (*MMAS*), que difere do AS em três aspectos principais: (I) apenas uma única formiga (a que encontrou a melhor solução na iteração atual ou aquela que encontrou a melhor solução desde o início do teste) adiciona feromônio após cada iteração, (II) os valores da trilha de feromônios são limitados a uma intervalo $[\tau_{\min}, \tau_{\max}]$ e (III) trilhas de feromônio são inicializadas para serem $\tau_{\max}$. Desta forma, o *MMAS* combina uma melhor exploração das melhores soluções encontradas durante a pesquisa com um mecanismo eficaz para evitar a estagnação precoce da pesquisa [20] , e funciona muito melhor do que o AS.

Outro importante aprimoramento do AS aplicado para resolver o TSP é o Ant Colony System (ACS) [6] , [8] , [12] . Semelhante ao *MMAS* , o ACS acrescenta as trilhas de feromônio dos arcos no melhor percurso desde o início da trilha, no entanto, ele usa a regra proporcional pseudo-aleatória para selecionar a próxima cidade quando as formigas constroem os passeios. Além disso, o ACS usa mecanismo de atualização de trilhas de feromônio local e explora a *lista de candidatos*, uma estrutura de dados que fornece informações heurísticas locais adicionais. Testes em instâncias de benchmark indicam que essas medidas podem não apenas melhorar a qualidade das soluções, mas também reduzir o tempo de computação. Com base nas modificações mencionadas acima, os métodos de busca local, como 2-opt e 3-opt, são integrados aos algoritmos ACO [6] , [10] , [18] . Os resultados da computação mostram que a aplicação de métodos de busca local pode melhorar significativamente a qualidade da solução.

Marcin e Tony [16] combinaram o algoritmo genético (GA) e o ACS, e propuseram dois algoritmos híbridos para melhorar o desempenho do ACS. Embora seu primeiro algoritmo que utiliza uma população de formigas genéticas modificadas por um AG, chamado ACSGA TSP, não possa superar o algoritmo ACS, tem a vantagem de convergência rápida e baixa variabilidade. O segundo algoritmo em [16] , denominado algoritmo Meta ACS-TSP, usa o GA para otimizar as configurações de parâmetros usadas no algoritmo ACS. Experimentos usando seu segundo algoritmo resultam em uma nova série de valores de parâmetros apropriados ao invés daqueles usados por Dorigo e Gambardella [6] . Um trabalho semelhante que usa o GA para otimizar as configurações dos parâmetros do ACS pode ser encontrado em [11] . Além disso, Hao e cols.[21] introduziu uma nova estratégia de controle de parâmetros adaptativos que usa otimização de enxame de partículas (PSO) para resolver o problema de definição de parâmetros.

Mais recentemente, Tsutsui [22] propôs uma variante do algoritmo ACO chamada de Ant System (cAS). Em cAS, uma formiga constrói uma solução de acordo com a densidade de feromonas e uma parte de uma solução de uma iteração anterior, e essa ação de astúcia pode reduzir a estagnação prematura e melhorar o desempenho do algoritmo ACO.

SEÇÃO III

O algoritmo proposto para o TSP

baixar PDF

Nesta seção, apresentamos o algoritmo proposto com o GA incorporado para o TSP. A idéia principal desse algoritmo é deixar que as formigas se consultem e, assim, aprimorar o mecanismo de compartilhamento de informações, o que pode ter mais chances de encontrar uma solução melhor. O operador de crossover do GA pode combinar os genes nos pais, e os descendentes do crossover podem ser vistos como o resultado da consultoria dos pais até certo ponto. Então, usamos o operador de crossover heurístico como método para as formigas consultarem os outros. O pseudo-código do algoritmo proposto é mostrado na Fig. 1 , onde $[i/2]$ representa o valor arredondado de $eu/2$. No algoritmo proposto, após inicializados os parâmetros que precisavam no algoritmo, $m$as formigas começam a construir as soluções que constituirão a população para GA. Então, o GA aprimorado embutido na ACO para otimizar as soluções construídas pelas formigas, e as trilhas de feromônio são atualizadas de acordo com S^{gb}(a melhor solução desde o início do julgamento). O programa será iterado até que a condição de finalização seja satisfeita. Na lembrança desta seção, apresentamos os principais operadores usados no algoritmo proposto.

A. A construção da solução

No algoritmo proposto, empregamos uma nova maneira gananciosa de construir soluções. Inicialmente, m formigas são colocadas em $m$cidades aleatoriamente. Então, em cada etapa de construção, cada formiga (na cidade E) tem que escolher uma cidade não visitada $j$como o próximo destino. No tempo t, uma formiga k que localizado na cidade Eseleciona a próxima cidade à luz das seguintes regras:

```
Procedure: The proposed algorithm
Begin
  Initialize parameters;
  While (termination condition not satisfied) do
    The ants construct m solutions which constitute the population P for GA;
    Evaluate the solutions;
    i = m;
    While (i >=2) do
      For (j=0; j< [i/2]; j++)
        Select two solutions, e.g. p1 and p2, from the population;
        Recombine the selected solutions using heuristic crossover operator
          to yield c(j);
        Mutate c(j) with mutation probability pm;
        Let the best solution in { p1, p2, c(j) } enters new_population;
      End for;
      i =[i/2];
      Replace the population with new_population;
    End while;
    Save the best solution found from the beginning as S^gb;
    Update the pheromone trails;
  End while;
End;
```

Figura 1 O algoritmo proposto em pseudocódigo.

1. ank primeiro escolhe a cidade designada da lista de candidato [17] da cidade E, a estrutura de dados estática que inclui *cl* cidades mais próximas da cidade E, e cada cidade $j$na lista de candidatos será selecionada com uma probabilidade [10] :

$$p_{eu}^{k}(j\,t) = \begin{cases} \dfrac{[\tau_{eu}(j\,t)]\cdot[\eta_{eu}]_{j}^{\beta}}{\sum_{k\in S_k}[\tau_{eu}(k\,t)]\cdot[\eta_{eu}]_{k}^{\beta}} & eu\ f\ j\in S_k \\ 0 & o\,t\,h\,e\,r\,w\,i\,s\,de\,e \end{cases}, \quad (1)$$

Ver fonte

Onde $\tau_{eu}(j\,t)$é o valor da trilha de feromônio na borda (i, j)no tempo t, $\eta_{eu}(j\eta_{eu}=1/d_{eu})$é a visibilidade de j de i, αe βsão parâmetros que determinam a importância relativa do rastro de feromônio e a visibilidade, e S_k é a intersecção da lista de

candidatos da cidade Eeu o conjunto de cidades que formiga k

ainda não visitou.

baixar PDF

2. E se $S_k = 1$ formiga k escolhe a próxima cidade j no conjunto de cidades que formiga k não visitou para passar para o seguinte a regra de transição de estado:

$$j = \arg \max_{s \in um\ l\ l\ o\ w'e\ d_k} \{ [\tau(i, s)]^{\alpha} \cdot [\eta(i, s)]^{\beta} \} \quad (2)$$

Ver fonte

Onde $um\ l\ l\ o\ w'e\ d_k$ é o conjunto de cidades que formiga k ainda não visitou.

B. O algoritmo genético incorporado

Em nosso algoritmo proposto, o AG é incorporado ao ACO com o objetivo de simular o processo de consulta entre formigas. om soluções construídas por formigas constituem a população inicial $P = \{ p_1, p_2, \ldots, p_m \}$ para GA. O tamanho da população será reduzido a metade de cada ciclo e o AG será interrompido até que apenas uma solução seja deixada na população. No GA incorporado, o valor de adequação de uma solução p_{Eu} é definido da seguinte forma:

$$fitness(p_{Eu}) = M - L_{Eu} \quad (3)$$

Ver fonte

Onde eu_{Eu} é o comprimento de p_{Eu} e M é definido como:

$$M = X \cdot \max_{Eu} \{ L_{Eu} \} \quad (i = 1, 2, \ldots, m) \quad (4)$$

Ver fonte

Onde λ é um coeficiente definido como 1,15 em nossos experimentos. Após a avaliação, as soluções serão selecionadas de acordo com a seleção da roleta para o operador de crossover.

1) Operador de crossover heurístico

Neste artigo, projetamos um operador de crossover heurístico aprimorado para produzir a descendência. O operador de crossover heurístico aprimorado utiliza informações de adjacência e informações heurísticas (incluindo distâncias e trilhas de feromônio entre as cidades) entre os genes para gerar a descendência.

Inicialmente, o operador crossover melhorado seleciona aleatoriamente uma cidade como a primeira cidade da descendência. Então o operador escolhe a cidade mais próxima entre o predecessor imediato e o sucessor da primeira cidade, chamada *vizinhos* , nos dois pais como a segunda cidade da prole. Se todas as cidades dos *vizinhos* foram visitadas, a próxima cidade será selecionada de acordo com a seguinte fórmula:

$$j = \arg \max_{k \in N} \{ fun(o_{Eu}, K) \} \quad (5)$$

Ver fonte

Onde N é o conjunto de cidades que não apareceram o. O processo é repetido até que todas as cidades sejam visitadas. Deixei n seja o número total de cidades, o ser prole e o_{eu-1} seja o eu th cidade em o. O pseudo-código do operador de crossover melhorado é mostrado na Fig. 2 .

2) Operador de mutação

Após o operador de crossover, os descendentes serão mutados com a probabilidade p_m para evitar a estagnação. No AG incorporado, usamos o operador de mutação de troca tripla em vez de 2-exchange descrito em [15] . O operador de 3 trocas seleciona aleatoriamente 3 cidades primeiro, e depois simplesmente troca os lugares delas. Obviamente,

existem 5 combinações possíveis além da prole em si, e a melhor delas

será selecionada como resultado da mutação.

baixar
PDF

C. Pheromone trail e sua atualização

Como o *MMAS* em [20] , o algoritmo ACO proposto impõe limites explícitos $\tau_{\min}$ e $\tau_{\text{máximo}}$ nas fugas mínima e máxima do feromônio para evitar a estagnação, e os valores iniciais de trilhas do feromônio são ajustados para ser $\tau_{\text{máximo}}$. Além do que, além do mais, $\tau_{\min}$ e $\tau_{\text{máximo}}$ são definidos da seguinte forma:

$$\tau_{\text{máximo}} = \frac{1}{1 - \rho} \cdot \frac{1}{f(S^{gb})} \quad (6)$$

$$\tau_{\min} = \tau_{\text{máximo}} / 2n \quad (7)$$

Ver fonte

Onde $0 < \rho < 1$ é a persistência da trilha, $f(S^{gb})$ é o custo da melhor solução globalmente e n é o número total de cidades. Depois que as soluções são otimizadas pelo GA, as trilhas de feromônio são atualizadas com base na melhor solução global (S^{gb}) De acordo com o *MMAS* proposto em [20] , a atualização do feromônio é feita da seguinte forma:

$$\tau_{eu}(t+1) = \rho\,\tau_{eu}(t) + \Delta\,\tau^{gb}_{eu\,j} \quad (8)$$

$$\text{where } \Delta\tau^{gb}_{eu\,j} = \begin{cases} 1/f(S^{gb}) & eu\ f(i,j) \in s^{gb} \\ 0 & otherwise \end{cases} (9).$$

Ver fonte

A equação (8) determina que apenas as arestas pertencentes à melhor solução global serão reforçadas.

Por fim, a condição de rescisão é definida quando o número de geração excede o número de geração permitido.

```
Procedure: The improved crossover operator
Begin
 i = 0;
 Select a random city as initial city o0 of the o;
 While ( i < n ) do
       Search the positions of oi in parents p1 and p2 respectively;
       If ( all the neighbors of oi in p1 and p2 have appeared in o)
        Select the next city j according to (5):
       Else
        Select the nearest city that has not appeared in o from the neighbors
          of oi as the next city j;
       End if;
       i = i+1;
       oi ← j;
 End while;
 on ← o0;
End;
```

Figura 2 O operador de crossover proposto no pseudocódigo.

SEÇÃO IV.
Resultados e análises computacionais

Nesta seção, apresentamos resultados numéricos para nosso algoritmo proposto e os comparamos com resultados de algoritmos ACO anteriores, incluindo o *MMAS* , ACS, Meta ACS TSP e cAS. Para o algoritmo proposto, vários parâmetros devem ser definidos. Nos experimentos a seguir, as configurações de parâmetros usadas são: $\rho = 0{,}8$, $\alpha = 1$, $\beta = 2$, $m = 35$, $p_m = 0{,}1$, $n_c = 20$. Como os valores para diferentes parâmetros são definidos heuristicamente, realizamos alguns testes adicionais e verificamos que, dentro de uma faixa moderada, não há diferença significativa nos resultados. Todas as

instâncias do TSP usadas aqui são obtidas da biblioteca de referência do TSPLIB.

baixar PDF

A. Desempenho do algoritmo ACO com o GA incorporado

Os resultados computacionais do algoritmo proposto são apresentados na Tabela I e os cálculos são executados por 1000 iterações. Na Tabela I , *opt* indica o valor da solução ótima conhecida de cada instância, mostra *melhor* o comprimento da melhor solução encontrada junto com seu desvio do ótimo conhecido em porcentagem, *média* e *pior* exibem a mesma informação para a média e a pior de 20 execuções independentes (10 execuções independentes para as 3 instâncias maiores). Na Tabela I podemos ver que o algoritmo proposto pode encontrar as soluções ótimas conhecidas de instâncias de benchmark menores testadas aqui em pelo menos uma das execuções. Tanto a *média*e os *piores* desvios do ótimo são inferiores a 1%. Além disso, para as instâncias kroA100 e lin105, todos esses 20 experimentos encontraram as soluções ótimas.

Na Tabela II , comparamos os resultados médios do algoritmo proposto com aqueles obtidos por *MMAS* , ACS e cAS. Os resultados do MMAS são retirados de [20] , e os de ACS, Meta ACS-TSP, cAS são de [12] , [16] e [22], respectivamente. A Tabela II mostra que os resultados obtidos pelo algoritmo proposto são melhores que, pelo menos, os obtidos por *MMAS* , ACS, Meta ACS TSP e cAS.

TABELA I Os resultados do algoritmo proposto são executados em instâncias do TSP

Instância	optar	melhor	média	pior
ei151	426	426 (0,0%)	426,20 (0,047%)	427 (0,235%)
ei176	538	538 (0,0%)	538,20 (0,037%)	539 (0,186%)
kroA100	21282	21282 (0,0%)	21282,00 (0,0%)	21282 (0,0%)
lin105	14379	14379 (0,0%)	14379,00 (0,0%)	14379 (0,0%)
ch130	6110	6110 (0,0%)	6121,95 (0,196%)	6155 (0,736%)
d198	15780	15781 (0,006%)	15800,25 (0,128%)	15826 (0,292%)
1em318	42029	42029 (0,0%)	42125,30 (0,229%)	42163 (0,319%)
pcb442	50778	50919 (0,278%)	50944,10 (0,327%)	50976 (0,390%)
att532	27686	27858 (0,621%)	27909,30 (0,807%)	27962 (0,997%)

TABELA II A comparação de resultados experimentais com outros algoritmos

Instância	proposta algoritmo de um	MMAS [b]	ACS [b]	Meta ACS-TSP [c]	cAS [b]
ei151	426,2	427,6	428,1	428,5	426,2
ei176	538,2	-	-	542,5	-
kroA100	21282,0	21320,3	21420,0	21513	21282,0
d198	15800,25	15972,5 I	16054,0	-	15954.1

Entre esses algoritmos, o Meta ACS TSP utilizou GA para otimizar os parâmetros da ACO. Esta é uma maneira diferente de combinar o ACO e o GA. A comparação na Tabela II demonstra a superioridade do nosso método. Outra coisa importante que deve ser destacada é que nosso algoritmo encontra esses melhores resultados através de muito menos iterações (os tempos de computação não são dados nessas literaturas).

B. A influência do GA embutido no desempenho do algoritmo ACO

Para ilustrar o papel do GA embutido, comparamos os processos evolutivos da melhor solução obtida pelo algoritmo proposto e o algoritmo sem o GA embutido na Fig. 3 . A Figura 3 mostra claramente que o GA embutido pode acelerar significativamente o processo de busca e melhorar a qualidade das soluções finais.

Assim, estes resultados sugerem fortemente que o GA embutido simula com sucesso o mecanismo de consulta entre formigas, e assim melhora o desempenho do algoritmo ACO.

SEÇÃO V.

Conclusão

baixar PDF

Neste artigo, um algoritmo ACO melhorado com GA embutido para o TSP é proposto. Dentro do algoritmo proposto, o algoritmo ACO é usado para construir soluções que constituem a população inicial para o algoritmo genético, enquanto o GA embutido é usado para simular o mecanismo de consulta entre as formigas para otimizar as soluções. No GA embutido, projetamos um novo operador de crossover heurístico que utiliza informações de adjacência, distâncias e trilhas de feromônio entre cidades para gerar descendentes. Os experimentos em instâncias de referência do TSP mostraram que o GA embutido pode melhorar significativamente o desempenho do ACO, e o algoritmo proposto supera os algoritmos ACO existentes para o TSP.

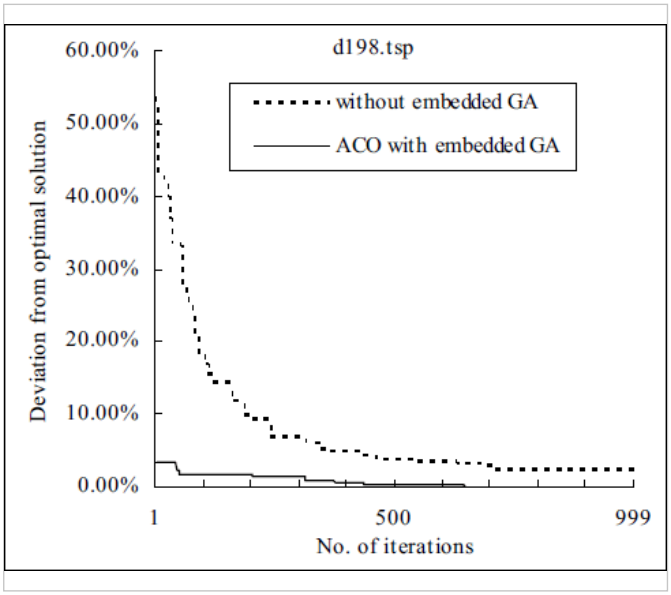

Fig. 3 Comparação do processo evolutivo entre o algoritmo proposto e o algoritmo sem o GA embutido.

Autores

Figuras

Referências

Citações

Palavras-chave

Métricas

IEEE Account

Profile Information

Purchase Details

Need Help?

Other

A not-for-profit organization, IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.
© Copyright 2019 IEEE - All rights reserved. Use of this web site signifies your agreement to the terms and conditions.

US & Canada: +1 800 678 4333
Worldwide: +1 732 981 0060

baixar PDF

Conta IEEE

» Alterar nome de usuário/Senha
» Atualizar endereço

Detalhes da compra

» Opções de pagamento
» Histórico de pedidos
» Visualizar documentos comprados

Informação do Perfil

» Preferências de Comunicações
» Profissão e Educação
» Interesses técnicos

Preciso de ajuda?

» **EUA e Canadá:** +1 800 678 4333
» **Em todo o mundo:** +1 732 981 0060
» Contato e Suporte

Sobre o IEEE *Xplore* | Contate-Nos | Socorro | Acessibilidade | Termos de uso | Política de Não Discriminação | Mapa do Site | Privacidade e exclusão de cookies

Uma organização sem fins lucrativos, o IEEE é a maior organização profissional técnica do mundo dedicada ao avanço da tecnologia para o benefício da humanidade.
© Copyright 2019 IEEE - Todos os direitos reservados. O uso deste site significa sua concordância com os termos e condições.