Algoritmo Time Colony Formiga com Algoritmos Genéticos

2 Autor (es) Hong-hao Zuo ; Fan-lun Xiong **Visualizar todos os autores**

Abstract: Time ant colony algorithm has good effect on combinatorial optimization problems as well as that of the ant colony algorithm while it has the shortcoming of long converge... **View more**

Metadata

Abstrato:
O algoritmo de colônia de formigas temporal tem um bom efeito sobre problemas de otimização combinatória, bem como o do algoritmo de colônia de formigas, embora tenha a desvantagem de um longo tempo de convergência. Um novo método combinado com algoritmos genéticos é proposto. Primeiramente, um procedimento de algoritmos genéticos é usado para resolver o problema especificando o tempo. Em segundo lugar, a solução obtida é usada para distribuir o feromônio original. No último momento, o algoritmo de colônia de formigas é usado para pesquisar a solução ótima, que supõe que a velocidade de cada formiga é a mesma e todas as formigas estão engatinhando em tempo integral. O novo método acelera a velocidade de convergência. É demonstrado pelo experimento que o novo algoritmo é melhor que antes

Publicado em: 2006 Conferência Internacional IEEE sobre Aquisição de Informações

Data da Conferência: 20 a 23 de agosto de 2006

Data adicionada ao IEEE *Xplore* : 12 de fevereiro de 2007

Informação ISBN:

Número de Acesso INSPEC **:** 9461182

DOI: 10.1109 / ICIA.2006.305886

Editora: IEEE

Localização da Conferência: Weihai, China

Contents

SEÇÃO I.
Introdução

Algorithm de Colônia de formigas (ACA) [1] , [2] é um tipo de algoritmo de computação de evolução baseado em biônica, que tem o caráter de feedback positivo e processamento paralelo. Foi proposto pela primeira vez para resolver o famoso problema do vendedor ambulante (TSP) [3] , [4] . Agora ele foi usado com sucesso para resolver todos os tipos de

problemas de otimização de combinação, como atribuição quadrática, programação de job-shop e assim por diante. Embora o ACA tenha sido usado com sucesso em muitos campos, é inconveniente para os novatos porque muitos parâmetros precisam ser ajustados em sua aplicação. Algoritmo de colônia de formigas temporais (TACA) [5]é apresentado e é uma versão melhorada do ACA. Supõe-se que a velocidade de cada formiga é a mesma e todas as formigas estão engatinhando em tempo integral. As formigas se comunicam com os outros pelo feromônio deixado na estrada. É demonstrado pelo experimento que o novo algoritmo é tão bom quanto o outro algoritmo de colônia de formigas e é mais simples justificar os parâmetros do que antes.

Algoritmos genéticos (GA) tem a capacidade de rápida pesquisa global, enquanto não pode usar as informações de feedback do sistema. Este personagem leva muitas iterações de redundância inúteis e baixa eficiência. TACA tem o caráter de distribuição, convergência paralela e global pela acumulação e atualização de feromônio. Sendo a falta de informação sobre feromonas durante o período de início, muitas vezes tem um longo tempo de pesquisa.

Para superar a limitação de cada algoritmo e obter vantagem mútua, é proposto um novo método. Primeiramente, primeiramente, um procedimento de algoritmos genéticos é usado para resolver o problema especificando o tempo, que tem o caráter de pesquisa global rápida. Em segundo lugar, a solução obtida é usada para distribuir o feromônio original. No último TACA é usado para procurar a solução ideal, que tem o caráter de feedback positivo e paralelo. O novo método é melhor que o TACA em eficiência de tempo, que pode encontrar uma solução melhor para especificar o tempo.

Está disposto abaixo: Na seção II é descrita a cronologia da colônia de formigas e alguns caracteres do algoritmo são narrados. Na seção III, os algoritmos genéticos usados no novo método são introduzidos. Na seção IV, o TACA com algoritmo genético é introduzido. Na seção V a prova da convergência do novo método dá-se. Na seção VI os resultados experimentais comparativos e a conclusão são dados.

SEÇÃO II
Algoritmo Time Colony Formiga

De acordo com o pensamento do processo real de forrageamento, isto é, todas as formigas estão trabalhando em tempo integral, um novo algoritmo de colônia de formigas baseado no modelo de tempo é proposto, no qual todas as formigas estão rastejando em tempo integral. Ele foi usado com sucesso no n-cities TSP.

A. Procedimento Básico

A hipótese deste algoritmo é a seguinte:

$D = \{d_{ij} | C_i, C_j \in c, 1 \leq i, j \leq n\}$ é o conjunto de distâncias entre cidades e n é o número de cidades. A distância mais curta do conjunto D é marcada como d_{min}. Todas as formigas estão engatinhando em todos os tempos e sua velocidade é igual a d_{min} por unidade de tempo. O tempo de decisão e assim por diante é omitido. Cada formiga viaja a mesma distância em cada ciclo que é sempre d_{min} ter T um b u $_{ij}$ salva as cidades que a formiga numerada já visitei. $\tau_{ij}(t)$ salva a densidade de feromônio na rota entre a cidade i e a cidade j no horário t.

O pseudo-código é como abaixo:

1. Inicialização, distribua as formigas nas cidades e adicione a cidade que a formiga localizar ao seu TabuList correspondente $_i$, defina todos $\tau_{ij}(0) = c$, Onde $1 \leq i, j \leq n$ e c é uma constante.

2. Ciclo a operação como abaixo, enquanto o resultado não foi convergência ou a contagem de ciclo é menor que o número dado:

3. Faça a operação como abaixo para toda a formiga k, onde $1 \leq k \leq m$, mis o número de formigas:

   Se (k numerado pela formiga pode alcançar a cidade i quando o ciclo atual terminar)

   Alcançar a cidade i // reachcity (i)

   Outro

   Observe a distância entre a formiga e a cidade i quando este ciclo terminar;

4. Atualizar todo o feromônio em toda a rota de acordo com o coeficiente volátil,

$$\tau_{eu}(t) = (1 - \rho) \cdot \tau_{eu}(T - 1), \quad (1)$$

Ver fonte

   ciclo para o passo 2;

5. Emita o resultado e o programa termina.

O pseudo-código de alcance da função (i) é descrito abaixo:

1. Adicione a cidade i à lista de rotas TabuList_k.

2. Calcular o comprimento do caminho que foi visitado, ou seja,

$$\text{Custo}_k(t) = d_{eu} + \text{Custo}_k(T - 1), \quad (2)$$

Ver fonte

   onde $\text{Custo}_k(t)$ é o custo da formiga numerada k no tempo t.

3. Libere o feromônio de quantidade fixa na rota entre as duas últimas cidades visitadas, nCurrentCity e nNextCity.

$$\tau_{eu}(t) = C + (1 - \rho) \cdot \tau_{eu}(T - 1), \quad (3)$$

Ver fonte

   onde C é uma constante.

4. Faça a decisão if-else conforme abaixo:

```
If(the ant's tour has finished(all the cities have been visited))
{
    prepare for the next tour, clear the route list TabuList_k, add city i to TabuList_k;
    set the distance which have traveled zero, Cost_k(t)=0;
}
else
    There is no operation.
```

   Não há operação.

5. Escolha a próxima cidade nNextCity de acordo com a probabilidade de transição de estado $P_{eu}^{k}(t)$, que será percorrido no próximo ciclo.

6. A formiga move-se para a cidade alvo no tempo restante deste ciclo e a distância entre a formiga e a cidade nNextCity é observada quando este ciclo termina.

7. Função reachcity (i) termina.

No algoritmo, a probabilidade de transição de estado $P_{eu_j}^k(t)$ é calculado da seguinte forma:

$$P_{eu_j}^k(t) = \begin{cases} \frac{[\tau_{eu_j}(t)]^\alpha}{\sum_{k \in umallowed_k}[\tau_{eu_k}(t)]^\alpha} & eu\ f\ j \in umallowed_k \\ 0 & otherwisdee \end{cases} \quad (4)$$

Ver fonte

B. Análise da Complexidade do Algoritmo

A complexidade do passo 1 é $O(n^2 \cdot m)$. A complexidade da etapa 3 é $O(m \cdot n^2)$. A complexidade do step4 é $O(n^2)$. Se o algoritmo for concluído após o ciclo NC vezes, a complexidade de tempo de todo o algoritmo é $O(NC \cdot m \cdot n^2 + NC \cdot n^2)$. Como o n é comparativo com m, a complexidade de tempo do algoritmo é $O(NC \cdot n^3)$, que é o mesmo que o algoritmo básico de colônia de formigas (abrev. como BACA).

C. Explique nos Parâmetros

Os parâmetros utilizados no BACA são o número de formigas me α, β, η relacionado à probabilidade de transição e ao coeficiente volátil ρe quantidade total Q relacionada ao feromônio [1]. Há menos parâmetros usados no novo algoritmo de colônia de modelo de tempo do que o de BACA, que é o número de formigasm, αrelacionado à probabilidade de transição e ao coeficiente volátil ρe constante C. Por causa de menos parâmetros, o novo algoritmo pode ser facilmente compreendido pelos novatos. A unidade de tempo não é a mesma que a da BACE, existe mais ciclo para a formiga chegar à próxima cidade do que antes. O feromônio sendo mais afetado, $\rho < 0,1$foi elogiado.

SEÇÃO III
Algorítmos genéticos

A forma usual de algoritmo genético foi descrita por Mitsuo Gen [6] como abaixo:

Procedimento: Algoritmos Genéticos

```
begin
  t←0;
  initialize P(t); %let P(t) be parents in current generation t
  evaluate P(t);
  while (not termination condition) do
    recombine P(t) to yield C(t);
     evaluate C(t);   %let C(t) be offspring
                      %in current generation t.
     select P(t+1) from P(t) and C(t);
     t←t+1;
  end
end
```

Algoritmos genéticos são técnicas de busca estocástica baseadas no mecanismo de seleção natural e genética natural. Algoritmos genéticos começam com um conjunto inicial de soluções aleatórias chamado população. Cada indivíduo da população é chamado de cromossomo, representando uma solução para o problema. Um cromossomo é uma cadeia de símbolos, que geralmente é uma cadeia de bits binários. Os

cadeia de símbolos, que geralmente é uma cadeia de bits binários. Os cromossomos evoluem por sucessivas iterações, chamadas gerações. Durante cada geração, os cromossomos são avaliados, usando algumas medidas de adequação. Para criar a próxima geração, novos cromossomos, chamados descendentes, são formados por (a) fusão de dois cromossomos da geração atual usando um operador de crossover ou (b) rejeição de outros para manter o tamanho da população constante. Cromossomos mais aptos têm maiores probabilidades de serem selecionados. Depois de várias gerações,

SEÇÃO IV.
Todo o procedimento de algoritmo

Existem algumas modificações na base do TACA e o novo método GATimeACA (GATACA) nasceu. Na primeira etapa, a função GADistribute é usada para distribuir o feromônio original. Os outros é o mesmo para o TACA. Todo o procedimento do algoritmo é como abaixo:

Procedimento GATIMEACA

1. Função GADistribute é usado para distribuir o feromônio original. distribua as formigas nas cidades e adicione a cidade que a formiga localiza ao seu TabuListi correspondente.

2. Ciclo a operação como abaixo, enquanto o resultado não foi convergência ou a contagem de ciclo é menor que o número dado:

3. Faça a operação como abaixo para toda a formiga k, onde $1 < k < m$, m é o número de formigas:

   Se (k numerado pela formiga pode alcançar a cidade i quando o ciclo atual terminar)

   alcançar a cidade i // reachcity (i)

   Outro

   observe a distância entre a formiga e a cidade i quando este ciclo termina;

4. Atualizar todo o feromônio em toda a rota de acordo com o coeficiente volátil,

$$r_{eu}(_{j}t) = (1 - \rho) \cdot \tau_{eu}(_{j}t - 1)$$

   Ver fonte

5. Emita o resultado e o programa termina.

O pseudo-código da função GADistribute () é descrito abaixo:

1. Inicialização: criar P_0 aleatoriamente, que é codificado em número real;

2. Avaliação: avaliar a população de geração n (P_n), a função do fitness é a recíproca do comprimento total do passeio;

3. Se a geração for igual a 30, vá para a etapa 9;

4. Selecione os indivíduos a serem cruzados, nos quais a cadeia cromossômica (i, j) e (i + 1, j + 1) são usados para substituir a sequência original (i, i + 1) e (j, j + 1);

5. Avalie a descendência esperada;

6. Selecione P (n + 1) de P (n) e C (n);

7. n = n + 1;

8. Ir para o passo 2;

9. um grupo de solução ótima é produzido, que são várias rotas de viagem;

10. As formigas rastejam na rota ótima e eliminam o feromônio C fixo nele. O feromônio original é distribuído.

$\tau_{ij} = 0$
for (m ants) do
{
$\tau_{ij} = C + \tau_{ij}$,which i,j is belong to the set of optimum route
}

11. Final da função.

SEÇÃO V.
Prova de Convergência

Imitando o método de [7] , a prova de convergência é dada abaixo:

V.
Proposição 1

Para qualquer τ_{eu} contém:

$$\lim_{\theta \to \infty} \tau_{eu}(j\theta) < \tau_{\text{máximo}} \frac{m\,C}{\rho}$$

Ver fonte

V.
Prova

A quantidade máxima possível de feromônio adicionada a qualquer arco (i, j) após qualquer iteração é $q_f(s^*)$ Neste algoritmo $q_f(s^*)$ é igual a mC, o que significa que todas as formigas passaram pelo arco (i, j) e atingiram a cidade numerada j. Claramente, na iteração 1, a trilha máxima possível de feromônios é

$$\tau_{eu}(j1) = (1 - \rho)\,\tau_0 + Q(1) \le (1 - \rho)\,\tau_0 + q_f(s^*) \quad (5)$$

Ver fonte

onde Q (1) é a quantidade de feromônio adicionada ao arco (i, j) após a iteração 1.

Na iteração 2 é

$$\tau_{eu}(j2) = (1 - \rho)\,\tau_1 + Q(2) \le (1 - \rho)\,\tau_1 + q_f(s^*) \quad (6)$$
$$\le (1 - \rho)^2\tau_0 + (1 - \rho)\,q_f(s^*) + q_f(s^*), \quad (7)$$

Ver fonte

onde Q (2) é a quantidade de feromônio adicionada ao arco (i, j) depois da iteração 2 e assim por diante. Assim, devido à evaporação do feromônio, a trilha do feromônio na iteração θ é limitado por

$$\tau_{eu}(_{j}\theta\ ) \ = \ (\ 1\ -\ \rho\ )\ \tau_{eu}(_{j}\theta\ -\ 1\ )\ +\ Q\ (\ \theta\ -\ 1\ ) \tag{8}$$

$$\leq\ (\ 1\ -\ \rho\ )_{T_{eu}}(\ \theta\ -\ 1\ )\ +\ q_{f}(\ s^{*}) \tag{9}$$

$$\leq\ (\ 1\ -\ \rho\ )^{\theta}\tau_{0}\ +\ \sum_{eu\ =\ 1}^{\theta}(\ 1\ -\ \rho\ )^{\theta\ -\ i}q_{f}(\ s^{*}) \tag{10}$$

$$\mathrm{L\ e\ t} \qquad \text{máximo} \lim_{\theta\ \rightarrow\ \infty}(\ 1\ -\ \rho\ )^{\theta}\tau_{0}\ +\ \sum_{eu\ =\ 1}^{\theta}(\ 1\ -\ \rho\ )^{\theta\ -\ i}q_{f}(\ s^{*}) \tag{11}$$

Ver fonte

Como $0\ <\ \rho\ \leq$**, então**

$$\lim_{\theta\ \rightarrow\ \infty}(\ 1\ -\ \rho\ )^{\theta}\tau_{0}\ =\ 0\ , \tag{12}$$

$$\lim_{\theta\ \rightarrow\ \infty}\sum_{eu\ =\ 1}^{\theta}(\ 1\ -\ \rho\ )^{\theta\ -\ i}\ =\ {}^{1}\!/_{\rho}, \tag{13}$$

Ver fonte

a equação 16 converge assintoticamente para

$$\tau_{\text{máximo}}\ \frac{q_{f}(\ s^{*})}{\rho}. \tag{14}$$

Ver fonte

Enquanto no algoritmo é impossível que a quantidade de feromônio adicionada a qualquer arco $(\ i\ ,\ j)$ **depois de cada iteração é sempre** $q_{f}(\ s^{*})$**, então de (9) para (10) o token = deve ser excluído.**

Isso é

$$\lim_{\theta\ \rightarrow\ \infty}\tau_{eu}(_{j}\theta\ )\ <\ \tau_{\text{máximo}}\ \frac{m\ C}{\rho}. \tag{15}$$

Ver fonte

Prova final.

V. Teorema

Deixei $P^{*}(\ \theta\ )$ **a probabilidade de que o algoritmo encontre uma solução ótima pelo menos uma vez dentro das primeiras** θ **iterações. Então, por um arbitrariamente pequeno** $\mathcal{E}\ >\ 0$ **e por um suficientemente grande** θ **afirma que**

$$P^{*}(\ \theta\ )\ \geq\ 1\ -\ \mathcal{E}.$$

Ver fonte

e, por definição,

$$\lim_{\theta\ \rightarrow\ \infty}P^{*}(\ \theta\ )\ =\ 1$$

Ver fonte

V. Prova

Devido aos limites de trilha de feromônio e $\tau_{\mathrm{m\ i\ n}}$ **e** $\tau_{\text{máximo}}$**, podemos garantir que qualquer escolha viável em (9) para qualquer solução parcial** x_{h} **é feito com uma probabilidade** $p_{\min}\ >\ 0$ **Um limite inferior trivial para** $p_{\min}$ **É dado por**

$$p_{\min}\ \geq\ \hat{p}_{\min}\ =\ \frac{\tau_{\min}^{\alpha}}{(\ n\ -\ 1\ )\ \tau_{\text{máximo}}^{\alpha}\ +\ \tau_{\min}^{\alpha}}, \tag{16}$$

Ver fonte

onde n é a cardinalidade do conjunto C de componentes (para a derivação deste limite consideramos a seguinte situação de "pior caso": a trilha de feromônio associada à decisão desejada é $\tau_{\min}$, todas as outras escolhas viáveis - há no máximo n-1 - têm uma pista de feromônio associada $\tau_{\text{máximo}}$). Então qualquer solução genérica s ', incluindo qualquer solução ótima $s^* \in S^*$ (S^* é um conjunto não vazio de soluções ótimas), pode ser gerado com uma probabilidade $\hat{p} \geq \hat{p}_{\min} > 0$. Onde $n < +\infty$ é o comprimento máximo de uma sequência. Porque é suficiente que uma formiga encontre uma solução ótima,

$$P^*(\theta) = 1 - \prod_{eu=1}^{\theta} (1 - \hat{p}_{Eu}) \tag{17}$$

$$\geq 1 - (1 - \hat{p}_{\min})^{\theta} = \hat{P}^*(\theta) \tag{18}$$

Ver fonte

Por causa de $1 > \hat{p}_{\min} > 0$, escolhendo um suficientemente grande θ essa probabilidade pode ser maior que qualquer valor $1 - \mathcal{E}$. Por isso, temos que

$$\lim_{\theta \to \infty} \hat{P}^*(\theta) = 1, \tag{19}$$

Ver fonte

então

$$\lim_{\theta \to \infty} P^*(\theta) = 1. \tag{20}$$

Ver fonte

Prova final.

Tabela 1 Resultado da Experiência

Algorithm \ Datasets		CTSP31	CHN144	CHN150	GR666
BACA	Average of 20 experiments	4.2264	30472.75	6558.9	294763.7
	Optimization found	4.1632	30355.12	6527	294358
TACA	Average of 20 experiments	4.2009	30450.00	6552.95	294698.8
	Optimization found	4.1662	30355.12	6527	294358
GATACA	Average of 20 experiments	4.1945	30435.83	6548.8	294631.5
	Optimization found	4.1588	30355.12	6527	294358

SEÇÃO VI.
Experiência e Conclusão

Para testemunhar a validade do novo algoritmo, foi realizada uma experiência especial com o TSP. Na Fig. 2 a distância entre duas cidades vizinhas é 10 e a distância entre duas cidades pode ser calculada pela geometria na qual (a) é para TSP de 9 cidades, (b) é para TSP de 16 cidades e (c) é para 25 cidades TSP. Posteriormente, o novo algoritmo foi utilizado para resolver o problema: CTSP3 [8] , [9] , CHN144 [10] , CHN150 [11] TSP225 [11] e GR666 [11] . O resultado está na tabela 1 [12] , [5] .

O resultado mostra que o desempenho do TACA é tão bom quanto o do BACA. Sendo mais uma ocasião para encontrar a otimização global, TACAE e GATACA foram utilizados com sucesso em vários TSP e seu desempenho é melhor que o de BACA e TACA. É testemunhado como um novo método de otimização eficaz. O trabalho adicional é a sua aplicação em simulação em tempo real e computação paralela.

RECONHECIMENTO

Zuo HH agradece ao seu colega Sa Li, Bai Shilei e Ding Jing por discutir algumas questões sobre este artigo.

Download PDF

Autores

Referências

Citações

Palavras-chave

Métricas

Conta IEEE

Informação do Perfil

Detalhes da compra

Preciso de ajuda?

De outros