به نام خدا

Ant colony optimization

نویسنده

لیلی صدیق،محمد نحوی

کلمات کلیدی

کلونی مورچه ها، مسئله فروشنده دوره گرد، بهینه سازی.

چکیده

در این مقاله نحوه بهینه سازی الگوریتم ها و یافتن سریع ترین راه حل ها با استفاده از روش کلونی مورچه ها بررسی شده است

یک منبع امید بخش تکرار رفتار در مورچه ها می باشد .برای پیدا کردن غذا به صورت رندم فضای اطراف آشیانه خود جستجو می کنند و وقتی غذا را پیدا کرده با توجه به نوع و کیفیت غذا مقداری از آن را به سمت لانه خود آورده و در حین برگشت از خود یک فرمون ترشح می کنند که از طریق این فرمون هم بقیه مورچه ها به سمت غذا آمده و هم با افزایش تعداد مورچه ها میزان فرمون ترشح شده افزایش پیدا کرده و کوتاهترین راه شناسایی می شود .

از این رفتار مورچه ها در حل بسیاری از مسائل بهینه سازی استفاده می شود .الگوریتم ACO بر اساس مدل جستجو کردن استوار(MBS) می باشد .الگوریتم MBS بسیاری از مسائل بهینه سازی مجزا را می تواند حل کند . الگوریتم MBS با استفاده ازاحتمال $M \in \mathcal{M}$ بیان می شود .که $\mathcal{M}$ تمام احتمال های ممکن است .

الگوریتم MBS به دو زیر مجموعه تقسیم می شود که در قسمت اول الگوریتم مدل احتمالی را می گیرد بدون اینکه در حین اجرا در ساختار آن تغییری ایجاد کند .اما در حالت دوم الگوریتم مدل احتمالی رادر فاز های مختلف تغییر می دهد . ACO یک الگوریتمی است که از حالت اول MBS استفاده می کند . در الگوریتم ACO مدل احتمالی مدل فرمونی(*pheromone model.*) نامیده می شود . مدل فرمونی شامل یه دسته از مدل های پارامتری $\mathcal{T}$ می شود که pheromone trail parameter نامیده می شود .pheromone trail parameter که $T_i \in \mathcal{T}$ دارای مقدار هستند که *pheromone values* نامیده می شوند .در هنگام اجرا الگوریتم ACO تلاش می کند تا مقدار فرمون ها را به روز کند تا راه حل را با کیفیت تر کند .مقدار فرمون ها با استفاده از مقدار های قبلی به روز می شود .

در اینجا یک چار چوبی برای الگوریتم ACO بیان می شود که hyper-cube framework (HCF) نامیده می شود و بر اساس قانون تغییر میزان فرمون ها در الگوریتم ACOاست .HCF منظر ساده تری از الگوریتم ACO می باشد.

AS اولین الگوریتم پیشنهادی برای ACO می باشد .

Solution Components and Construction Graph:

اگر به حداقل رساندن مساله را با $\mathcal{P} = (\mathcal{S}, f, \Omega)$ نشان بدهیم که $\mathcal{S}$ میباشد *set of candidate solutions* و f هست objective function که قابل ارجاع به هر یک از $s \in \mathcal{S}$ که $f(s)$ را ایجاد می کند و Ω هست *set of constraints* می باشد .

مسائل CO(combination optimization) شامل مراحل زیر می باشد :

CO را به صورت زیر نمایش می دهند :

$\mathcal{P} = (\mathcal{S}, \Omega, f)$ یک مدل برای CO می باشد . که در آن S فضای جستجو یا راه حل می باشد که یک مجموعه متناهی است از مقدار مجزا می باشد و Ω محدودیت بین مقادیر می باشد .

تابع $f : \mathcal{S} \to \mathbb{R}^+$ باید مینیمم شود.

فاصله دو شهر را با (i,j) نشان می دهند . فرمون بین دو شهر را با τ_{ij} نشان می دهیم

.

Ant colony system

این الگوریتم شامل یک فیدبک مثبت می شود .که این فیدبک مثبت که اطمینانی برای پیدا کردن یک راه حل سریع می باشد .محاسبه اشیا پخش شده از همگرایی نابهنگام جلوگیری می کند .این الگوریتم برای مشکل فروشنده سیار طراحی شده است (TSP).

ما برای حل این مشکل از مورچه ها استفاده کردیم که با هم همکاری کرده برای حل مشکلشان با استفاده از فرمون هایی که در لبه های گراف از خود ترشح می کنند . مورچه ها به صورت موازی به دنبال راه حل برای فروشنده سیار هستند.

در حالت کلی الگوریتم ACS می تواند تمام مسائل ترکیبی را حل کند . این الگوریتم به صورت زیر بیان می شود :

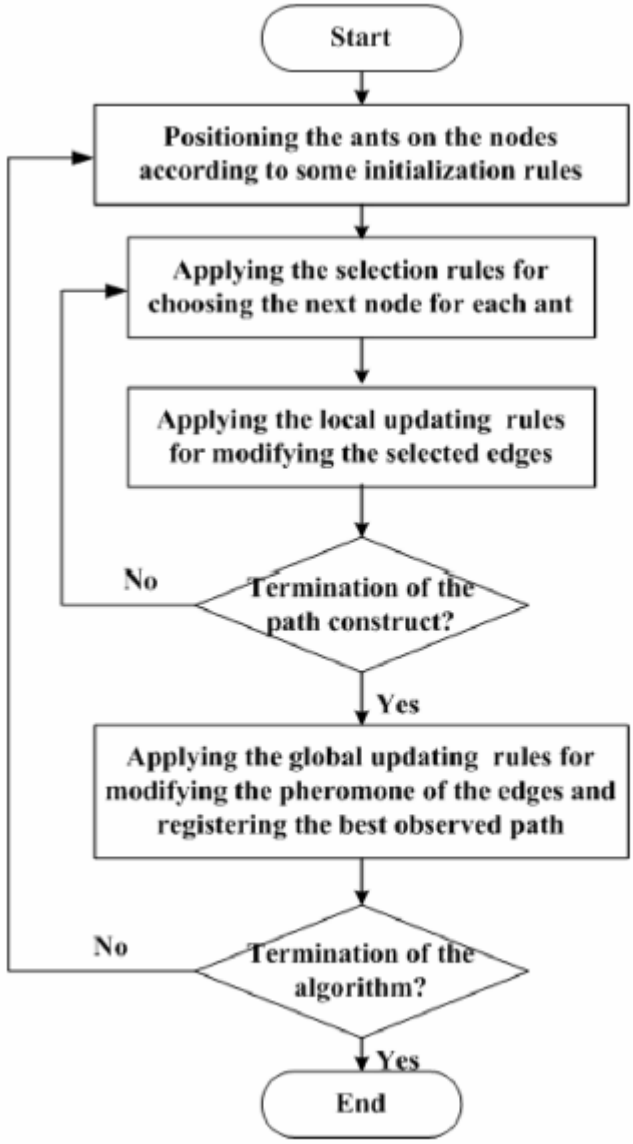
۱- در قسمت اول این الگوریتم مورچه ها به صورت رندم وتصادفی در روی راس های گراف قرار میگیرند . و یک مقدار اولیه برای به دنبال کشیدن روی لبه گراف قرار می دهند .

۲- در قسمت دوم این الگوریتم مورچه ها به صورت احتمالی به نقطه بعدی بر اساس قانون های احتمالی حرکت می کنند .مورچه ها روی کوتاهترین لبه حرکت می کنند و از قسمتی می روند که بیشترین فرمون را دارا هست . و این کار اینقدر ادامه پیدا می کند تا کل مورچه ها در این مسیر حرکت کنند .

۳- در مرحله بعد باید هرمورچه ارزیابی شود تا مسائل بهینه سازی به بهترین وجه صورت گیرد و میزان فرمون بین دو لبه باید به روز شود .

۴- فرمون های روی یک لبه با گذشت زمان تبخیر می شوند به همین دلیل باید این فرمون ها به روز شوند.

۵- کل این مراحل زمانی پایان می پذیرد که کل مورچه ها از یک مسیر در حال حرکت باشند .



Flowchart of ACS algorithm.

مسائل اصلی که توسط ACO حل می شود را می توان به صورت زیر دسته بندی کرد :

۱-الگوریتم تکاملی براساس ژنتیک و فرضیه تکاملی

۲-شبیه سازی حرارت بر اساس دینامیک حرارت

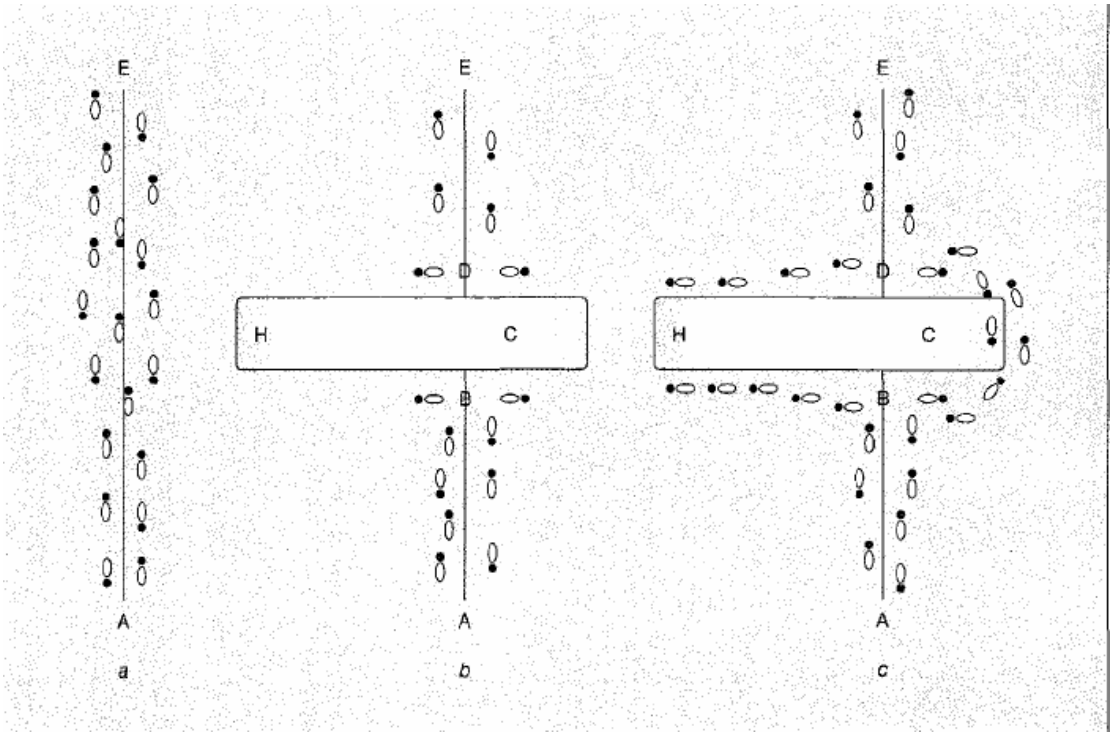
۳-تحقیقاتی که بر اساس حافظه کوتاه مدت انجام می شود و...

همانطور که گفته شد مورچه ها بر اساس فرمون هایی که از خود ترشح می کنند هم مسیر حرکت را دنبال کرده و هم از این طریق می توانند کوتاهترین راه را بین آشیانه و غذا پیدا کنند . این مراحل می توانند به صورت زیر بیان بشوند :

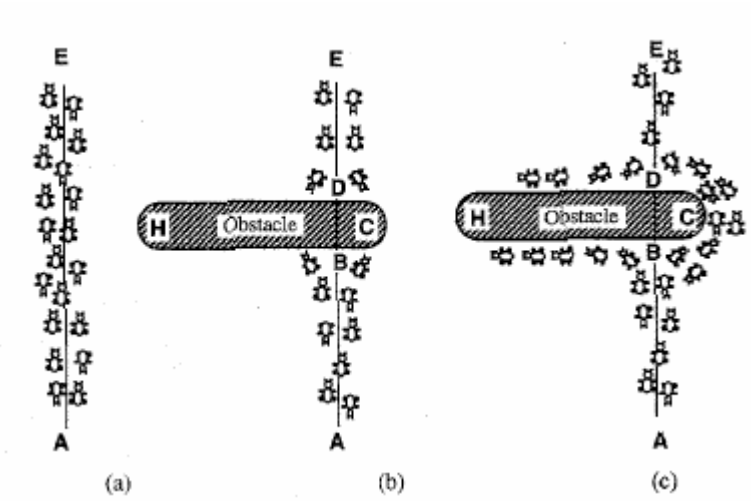
در شکل زیر در قسمت اول مورچه ها روی یک مسیر مستقیم که از آشیانه به غذا وصل می شود حرکت می کنند وقتی که یک مانع در سر راهشان ایجاد می شود مسیر حرکت قطع شده ، و به دنبال یک راه حل اکتشافی برای حل مشکل خود می گردند .

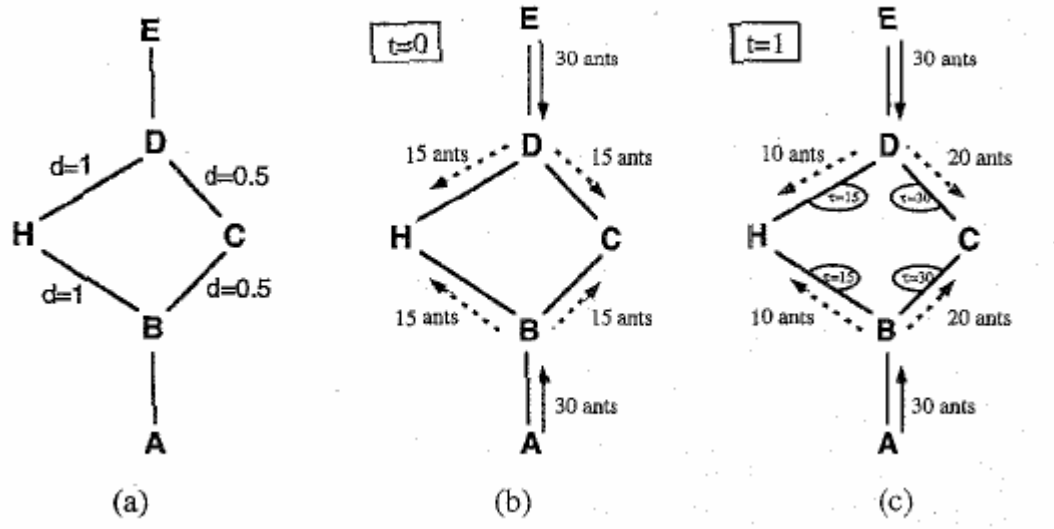
مورچه هایی که بالای مانع هستند نمی توانند مسیر حرکت را پیدا کنند زیرا مسیر فرمون را گم کرده اند به همین دلیل این احتمال را دارند که به سمت راست حرکت کنند یا به سمت چپ حرکت کنند . اما مورچه هایی که بر حسب تصادف مسیر کوتاهتر را انتخاب کرده اند زودتر می توانند این اتصال بین فرمون ها را فراهم کنند . به همین دلیل مسیر کوتاهتر می تواند سریعتر از فرمون پر شود و مورچه های بیشتری را به سمت خود بکشد . این فیدبک مثبت می تواند بسیاری از مورچه ها را به سمت خود بکشد .

مورچه های مشابه به هم از طریق فرمون با هم ارتباط برقرار می کنندو رفتار خود را بر اساس موقعیت تغییر می دهند .



شکل بالا را نیز می توان به صورت زیر بیان کرد و نقاط را نام گذاری کرد:





الگوریتم ant colony optimization را می توان به صورت زیر خلاصه کرد :

اولا باید یک گراف بین مسیر های حرکت غذا در نظر گرفت .

در ابتدا یک مقدار اولیه فرمون به هریک از اضلاع گراف نسبت داد و یک مورچه را به صورت تصادفی در مکان جستجو قرار داد. برای هر مورچه مراحل زیر باید انجام شود :

۱- قانون حرکت احتمالی: براساس این قانون مورچه را در فضای جستجو حرکت داده به این ترتیب راه حل مساله ایجاد می شود .

۲- ارزیابی بهترین : باید بهترین راه حلی که توسط این مورچه ایجاد شده را ارزیابی کرد .

۳- به روز کردن فرمون : فرمون هر ضلع را به روز می کنیم با استفاده از تقویت یک را حل خوب.

۴- دوباره به مرحله دوم برگشته و اینکار را ادامه داده تا به میزان فرمون دلخواه برسیم .

حال می توان به صورت زیر نیز بیان کرد :

اگر $\tau(r,s;t)$ شدت به دنبال کشیدن در روی ضلع (r,s) در زمان t باشد. وقتی که به روز شد در زمان t+1 رابطه زیر به دست می آید:

$$\tau(r,s;t+1) = \rho\tau(r,s;t) + \Delta\tau(r,s;t) \qquad (1)$$

ρ پارامتری است که $1-\rho$ ضریب تبخیر را نشان می دهد برای جلوگیری از انباشته شدن زیاد فرمون $\rho \in (0,1)$. تقویت به دنبال کشیدن مورچه ها شامل جمع شدن فرمونی است که هر یک تولید می کنند که رابطه اش به صورت زیر بیان می شود :

$$\Delta\tau(r,s;t) = \sum_{k=1}^{m} \Delta\tau_k(r,s;t) \qquad (2)$$

که تعداد مورچه ها در کلونی می باشد.

$\Delta\tau_k(r,s;t)$ مقدار فرمونی است که توسط k امین مورچه در زمان t باقی گذاشته شده است که به صورت زیر بیان می شود :

$$\Delta\tau_k(r,s;t) = \begin{cases} \dfrac{Q}{L_k}, & \text{if the } k\text{th ant uses the edge } (r,s) \\ 0, & \text{otherwise} \end{cases} \qquad (3)$$

که Q در آن یک مقدار ثابت می باشد و L_k فاصله سفر k امین مورچه است .

حال $\eta(r,s)$ میدان دید بین راس r و s را بیان می کند . $\eta(r,s)$ به صورت $\eta(r,s) = 1/w(r,s)$ که $w(r,s)$ وزن مقادیر روی لبه (r,s) هست که در طول مدت مراحل انجام شده تغییر نکرده است .

احتمال اینکه مورچه k انتخاب کند راس s را بعنوان راس بعدی وقتی که در راس r در زمان t قرار دارد به صورت زیر بیان می شود :

$$p_k(r,s;t) = \begin{cases} \dfrac{\tau^{\alpha}(r,s;t)\eta^{\beta}(r,s)}{\sum_{u \in M_k} \tau^{\alpha}(r,u;t)\eta^{\beta}(r,u)}, & \text{if } s \in M_k \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

M_k رئوسی هست که توسط مورچه k دیده نشده است . α و β پارامتر هایی هستند که میزان به دنبال کشیدن را در مقابل میدان دید کنترل می کنند.

الگوریتم ACO را به صورت زیر می توان نمایش داد:

```
1   Procedure ACO
2     Initialize parameters and
3     pheromone trails
4     for colony := 1 to ncolonies do
5       for ant := 1 to m do
6         Generate tour
7         Compute cost
8         if cost < min cost
9           Save solution
10          min_cost = cost
11        end if
12        Update pheromone trails
13      end for
14      Evaporate
15    end for
16  end Procedure
```

. ACO algorithm.

برای حل مشکل فروشنده سیار که باید از n شهر دیدن کند و از هر کدام فقط یک بار عبور کند میتوان ازالگوریتم ACO استفاده کرد که مراحل کار به صورت زیر می باشد :

اگر d_{ij} فاصله اقلیدسی بین شهر های i و j باشد که به صورت زیر قابل محاسبه هست:

$$d_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}).$$

اگر برای حل مشکل فروشنده سیار زوج مرتب (N, E)، که در آن N تعداد شهرها و E اضلاع گرافی هستند که تمام حالت های آن در نظر گرفته شده است .

$b_i(t)\,(i=1,\ldots,n)$ تعـــداد مورچـــه هـــا در شـــهر i هـــستند در زمـــان t و $m = \sum_{i=1}^{n} b_i(t)$ تعداد کل مورچه ها می باشد .

هریک از مورچه ها باید مراحل زیر را انجام دهد :

به صورت احتمالی به یکی از شهرها بروند که این احتمال تابع فاصله تا شهر و میزان فرمون موجود در مسیر است .

برای اینکه سفر مورچه قانونی باشد ،حق دیدن شهر را ندارد.

وقتی سفر مورچه تمام شد از خود ماده ای دفع می کند .

$\tau_{ij}(t)$ میزان قدرت به دنبال کشیدن روی ضلع (i,j) در زمان t می باشد .

الگوریتم AS توسط m مورچه m بار در فاصله زمانی $(t,t+1)$ تکرار می شود .

هر n بار تکرار الگوریتم AS که ما آن را یک حلقه می نامیم هر مورچه یک سفر کامل را انجام میدهد .

در این صورت قدرت به دنبال کشیدن طبق رابطه زیر به روز می شود :

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij} \qquad (1)$$

که ρ یک ضریب می باشد و $(1-\rho)$ میزان تبخیر بین زمان هایt و t+n می باشد.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (2)$$

در رابطه بالا $\Delta\tau_{ij}^{k}$ میزان فرمون در روی لبه (i,j) می باشد بوسیله k امـین مورچـه بـین زمان های t و t+n می باشد که به صورت زیر بیان می شود :

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if } k\text{th ant uses edge } (i,j) \text{ in its tour (between time } t \text{ and } t+n) \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

که در آن Q ثابت می باشد و L_k طول مسافتی است که k امین مورچه طی کرده اسـت و ρ یک مقداری کوچکتر از یک می باشد .

امکان انتقال مورچه k ام از شهر I به شهر j را می توان به صورت زیر نمایش داد:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}]^{\beta}} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

و β α پارامتر هایی هستند که میزان اهمیت به دنبال کشیدن در مقابل میدان دید کنترل می کند .

برای الگوریتم می توان موارد زیر را در نظر گرفت :

Formally the *ant-cycle* algorithm is:

1. Initialize:
   Set $t := 0$ {t is the time counter}
   Set NC $:= 0$ {NC is the cycles counter}
   For every edge (i, j) set an initial value $\tau_{ij}(t) = c$ for trail intensity and $\Delta\tau_{ij} = 0$
   Place the m ants on the n nodes

2. Set $s := 1$ {s is the tabu list index}
For $k := 1$ to m do
Place the starting town of the kth ant in $\mathbf{tabu}_k(s)$

3. Repeat until tabu list is full
{this step will be repeated $(n-1)$ times}
Set $s := s + 1$
For $k := 1$ to m do
Choose the town j to move to, with probability $p_{ij}^k(t)$ given by Eq. (4)
{at time t the kth ant is on town $i = \mathbf{tabu}_k(s-1)$}
Move the kth ant to the town j
Insert town j in $\mathbf{tabu}_k(s)$

4. For $k := 1$ to m do
Move the kth ant from $\mathbf{tabu}_k(n)$ to $\mathbf{tabu}_k(1)$
Compute the length L_k of the tour described by the kth ant
Update the shortest tour found
For every edge (i, j)
For $k := 1$ to m do

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{if } (i,j) \in \text{tour described by } \mathbf{tabu}_k \\ 0 & \text{otherwise} \end{cases}$$

$\Delta\tau_{ij} := \Delta\tau_{ij} + \Delta\tau_{ij}^k$;

5. For every edge (i, j) compute $\tau_{ij}(t+n)$
   according to equation $\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta\tau_i$
   Set $t := t+n$
   Set $\mathrm{NC} := \mathrm{NC}+1$
   For every edge (i, j) set $\Delta\tau_{ij} := 0$
6. If ($\mathrm{NC} < \mathrm{NC_{MAX}}$) and (not stagnation behavior)
   then
      Empty all tabu lists
      Goto step 2
   else
      Print shortest tour
      Stop

پیچیدگی الگوریتم *ant-cycle* به صورت $O(\mathrm{NC} \cdot n^2 \cdot m)$ است اگر ما الگوریتم را بعد از NC مرحله به پایان برسانیم . در حقیقت مرحله ۱ به صورت $O(n^2+m)$ ، مرحله ۲ به صورت $O(m)$ ، مرحله ۳ به صورت $O(n^2 \cdot m)$ ، مرحله ۴ به صورت $O(n^2 \cdot m)$ مرحله ۵ به صورت $O(n^2)$ و مرحله ۶ به صورت $O(n \cdot m)$ بیان می شود.

ما درحقیقت یک رابطه خطی بین تعداد مورچه ها و تعداد شهرها به صورت خطی پیدا کرده ایم .