Algorytmy mrówkowe - opis teoretyczny

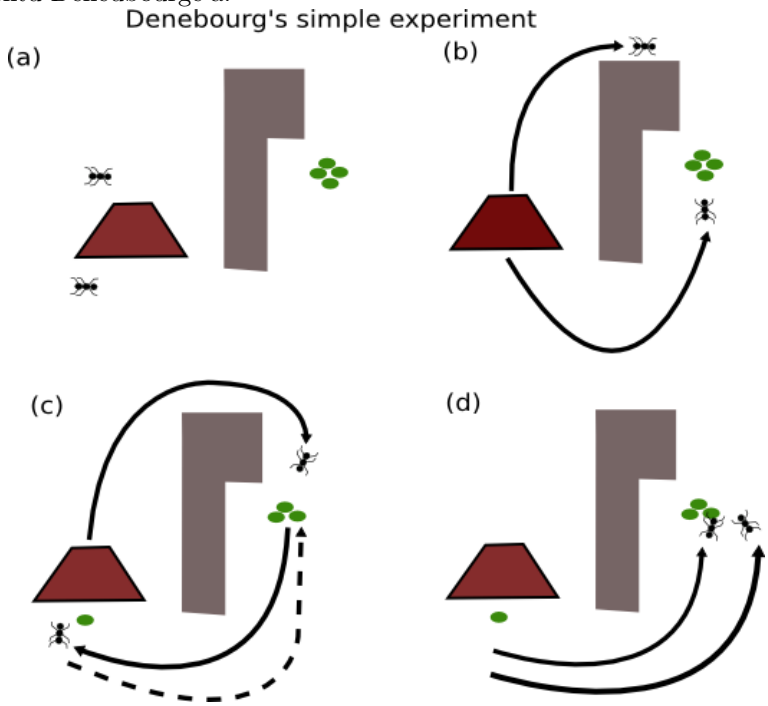
Tomasz Duszka (t.duszka@cyfronet.pl)
Jakub Janczak (j.janczak@cyfronet.pl)

11 czerwca 2007

1 Wprowadzenie

Poszukując pożywienia mrówki losowo wędrują po określonym obszearze pozostawiając za sobą ślad feromonowy. W momencie gdy inna mrówka trafi na taki ślad, to większe jest prawdopodobieństwo, że w kolejnym kroku wybierze drogę na której on się znajduje. Obierając miejsca już odwiedzone kolejne mrówki wzmacniają ślad, tym samym dając sygnał innym mrówkom o potencjalnie lepszej ścieżce do wyboru.

Ideę działania kolonii mrówek można zaobserwować na przykładzie eksperymentu Deneubourge'a:



Jak widać po dłuższej chwili (rys. (d)) mrówki mają tendencję do wybierania ścieżki krótszej.

Dzięki zjawisku parowania feromonu, mrówki ciągle dążą do znalezienia prawidłowego rozwiązania i unikają wpadania na zawsze w obiecujące, ale nie najlepsze rozwiązania. Poza tym im krótsza jest ścieżka tym podróż z mrowiska do pożywienia trwa krócej, co daje kolejną przewagę lepszym rozwiązaniom. Dzięki zastosowaniu pewnego stopnia losowości w wyborze następnego kroku, mamy pewność że część mrówek będzie poszukiwała nowych (możliwe że lepszych) rozwiązań.

Mając na uwadze prostotę zachowania kolonii mrówek, stwierdzono, że taki schemat działania może być bardzo dogodnym rozwiązaniem do przeprowadzania wszelkiego rodzaju optymalizacji. Wprowadzenie w życie ACO (Ant Colony Optimization) będących lekkimi modyfikacjami naturalnego zachowania mrówek, dało zaskakująco dobre wyniki wydajnościowe w tej klasie problemów.

1.1 Podobieństwa w stosunku do naturalnej kolonii mrówek

- modeluje kooperujące ze sobą mrówki
- wykorzystuje właściwości śladu feromonowego (parowanie, wzmacnianie)
- prowadzi do optymalnego rozwiązania
- wykorzystuje procesy stochastyczne przy wyborze następnego kroku

1.2 Odstępstwa od orginału

- wykorzystuje przestrzeń dyskretną jako modelowane środowisko
- przechowuje pamięć mrówek - insekty prawie jej nie posiadają
- często ilość pozostawianego feromonu zależy od jakości znalezionego rozwiązania
- w lepszych modelach feromon zostaje pozostawiony po przejści całej ścieżki
- dodatkowo stosuje optymalizacje niewystępujące w środowisku naturalnym:
  - przewidywanie ruchów w przód
  - wycofywanie się
  - lokalne optymalizacje poszukiwań (innym algorytmem)

2 Elementy algorytmu

2.1 Środowisko

Dyskretna przestrzeń - graf. Wierzchołki połączone są ścieżkami ważonymi (waga oznacza w tym przypadku długość ścieżki). Dodatkowo każda ze ścieżek przenosi informację o ilości pozostawionego feromonu.

Wprowadza się dwa oznaczone wierzchołki - mrowisko i wierzchołek z jedzeniem (dla uproszczenia jest tylko jeden wierzchołek z jedzeniem, a jego ilość jest nieskończona).

2.2 Cel

Znalezienie najkrótszej drogi (o najmniejszej sumarycznej wadze) od mrowiska do źródła jedzenia.

2.3 Podstawowy wzór

Jako podstawę algorytmu ACO przyjmuję się sposób wybierania następnego kroku. Wybór ten jest przeprowadzany na podstawie ilości mrówek które poprzednio odwiedziły dany wierzchołek i stały przed takim samym wyborem w poprzednich iteracjach. Jeśli za R_m i L_m przymiemy ilość mrówek które w danym punkcie odpowiednio wybrały ścieżkę L (lewą) i R (prawą) (poprzednio dany wierzchołek odwiedziło m mrówek, $R_m + L_m = m$. Prawdopodobieństwo, że mrówka $m + 1$ wybierze ścieżkę prawą jest równe:

$$P_R(m) = \frac{(R_m + k)^d}{(R_m + k)^d + (L_m + k)^d}$$

i odpowiednio dla lewej ścieżki (więcej ścieżek nie ma):

$$P_L(m) = 1 \quad P_R(m)$$

Parametry k i d dobieramy z góry, tak aby najlepiej oddawały rzeczywisty model (wartości wyznaczone symulacją Monte Carlo dla naturalnego mrowiska wynoszą odpowieniednio k 20, d 2).

Mając na uwadze powyższy wzór dobór ścieżki w powyższym przypadku można wyrazić jako zależność:

$$\begin{cases} R_m + 1 = R_m + 1, L_{m+1} = L_m & \text{dla } r <= P_R(m) \\ R_m + 1 = R_m, L_{m+1} = L_m + 1 & \text{dla } r > P_R(m) \end{cases}$$

gdzie r jest zmienna losową o rozkładzie ciągłym w przedziale $[0, 1]$. Powyższy przykład nie jest może zbyt skomplikowany, ale w pełni oddaje istotę wyboru następnego kroku.

2.4 Postępowanie z feromonem

Wyróżnia się dwa podejścia do postępowania z feromonem podczas optymalizacji:

1. Pozostawianie feromonu w trakcie przechodzenia przez graf
2. Pozostawianie feromonu po przejściu całej ścieżki

Drugi ze sposobów jest lepszy ponieważ nie zakłóca działania innych mrówek w iteracji, oraz (co ważniejsze) w trakcie dysponowania feromonem można wziąć pod uwagę jakość znalezionego rozwiązania.

Zasadniczym wzorem na ilość pozostawionego feromonu w chwili t jest:

$$_{ij}(t) = (1 \qquad _{ij}(t) +$$

gdzie to z góry ustalony parametr, a to zdefiniowany przyrost feromonu w czasie.

W konkretnych rozwiązaniach stosuje się lekkie jego modyfikacje.

2.5 Ogólny algorytm ACO

Ogólny algorytm ACO można wyrazić pseudokodem:

```
while(1) {
ants_generation_and_activity(); // (1)
phreomone_evaporation(); // (2)
daemon_actions(); // optional (3)
}
```

Kolejno kroki: (1) oznacza zarządzanie generowaniem mrówek i poruszanie już istniejących (2) to zarządzanie parowaniem feromonu poprzez uaktualnianie jego ilości na każdej ze ścieżek (zmniejszanie jego ilości) (3) opcjonalne działania demona, to dodatkowe działania niezależne od mrówek, takie jak na przykład zwiększanie ilości feromonu na najlepszych ścieżkach, czy wpływanie stanu jednych mrówek na inne (szczątkowa komunikacja)

3 Ant System

Jednym z pierwszych problemów który próbowano rozwiązać za pomocą algorytmów mrówkowych jest problem komiwojażera. Definiuje się go jako problem znalezienia najkrótszego cyklu Hamiltona w danym grafie pełnym $G =< V, E >$. Dla przypomnienia grafem pełnym nazywamy graf w którym każde dwa różne wierzchołki są połączone krawędzia. Cykl Hamiltona jest to natomiast cykl przechodzący przez każdy wierzchołek dokładnie jeden raz. Problem komiwojażera jest problemem NP-zupełnym, co oznacza, że większość naukowców sądzi, że nie da się go rozwiązać w czasie wielomianowym (ze wzlędu na wielkość grafu). Podstawowy algorytm mrówkowy rozwiązujący problem komiwojażera został

nazwany algorytmem AS (Ant System). Oczywiście jest to algorytm heurystyczny - nie mamy gwarancji, że znalezione rozwiązanie (cykl Hamiltona) jest najlepsze, ale że jest dość dobre. Dzięki temu algorytm ten działa bardzo szybko nawet dla większych rozmiarów grafów.

3.1 Opis algorytmu

Algorytm AS składa się z szeregu iteracji. W pojedyńczej iteracji dany zbiór mrówek wykonuje n kroków (n jest równa ilości węzłów w grafie) przemieszczając się pomiędzy węzłami. Wskutek przemieszczania się mrówek każda z nich buduje pewne rozwiązanie (pewien cykl Hamiltona) które jest podstawą do aplikacji feromonu w grafie. Wymyślono 3 główne sposoby w jaki uaktualniany jest feromon:

- gęstościowy (ant-density): mrówka w trakcie budowania drogi zostawia stałą ilość feromonu na krawędziach
- ilościowy (ant-quantity): mrówka w trakcie budowania drogi zostawia ilość feromonu odwrotnie proporcjonalną do dlugości krawędzi
- cykliczny (ant-cycle): feromon na krawędziach jest dodawany dopiero w momencie zbudowania całej drogi przez mrówke

Ponieważ początkowe testy wykazywały, że sposób cykliczny zachowuje się znacznie lepiej niż pozostałe sposoby, więc skupiono się głównie na nim w trakcie dalszych prac. Od tego momentu algorytm AS oznacza własnie algorytm mrówkowy dla problemu komiwojażera ze sposobem cyklicznym dodawania feromonu.

Feromony są przypisywane poszczególnym krawędziom typu (i, j) i oznaczają w pewien sposób prawdopodobieństwo tego, że mrówka z miasta i przejdzie do miasta j. Ilość feromonu jest dopierana zgodnie ze znalezionym rozwiązaniem. Im lepsze rozwiązanie znajdzie mrówka (krótszy koszt) tym więcej feromonu zostawia na krawędziach. Taki sposób zostawiania feromonu ma za zadanie nakierować przyszłe mrówki na juz znalezione dobre rozwiązania. Dodatkowo parowanie feromonu zapobiega tutaj stagnacji - stanowi kiedy wszystkie mrówki podążają tą samą drogą. Ponieważ aplikacja feromonu odbywa się dopiero po znalezieniu drogi przez mrówkę, więc musi ona być wyposażona w pewien obszar pamięci. Obszar ten nazywany jest z ang. "tabu list" i służy zapamiętywaniu kolejnych miast które odwiedza mrówka. Pamięc mrówki jest także wykorzystywana aby mrówka nie odwiedzała juz odwiedzonych miast (zgodnie z definicją cyklu Hamiltona który mrówka buduje).

Osobnym problemem pozostaje sposób wyboru kolejnych krawędzi przez mrówkę. Stosuje się w tym celu tzw. tablice decyzyjne które łączą ślad feromonowy oraz lokalne heurystyki wynikające z długości krawędzi. Tablice decyzyjne są dane wzorem:

$$a_{ij}(t) = \frac{\tau_{ij}(t) \quad \eta_{ij}}{\sum_l \tau_{il}(t) \quad \eta_{il}}$$

Gdzie i oraz j oznacza pewną krawędź, $_{ij}(t)$ to wartość feromonu na krawędzi i-¿j w czasie t, a n_{ij} to koszt krawędzi i-¿j. Wartości oraz to pewne parametery, które są dobierane zwykle w sposób eksperymentalny. Okreslają one balans pomiędzy podążaniem za feromonem a lokalnym wybieraniem zawsze najlepszej krawędzi. Dla przykładu przy wyborze $= 0$ oraz $= 1$ algorytm AS staje się zwykłym algorytmem zachłannym zawsze wybierającym najlepsze krawędzie.

Mając dane tablice decyzyjne możemy wyznaczyć prawdopodobieństwo wyboru określonej krawędzi:

$$p_{ij}(t) = \frac{a_{ij}(t)}{\sum_l a_i lt}$$

Ilość feromonu która jest dodawana do krawędzi po zbudowaniu drogi jest dana wzorem:

$$_{ij}(t) = 1/L$$

W powyższym wzorze L jest to długość znalezionej przez mrówke drogi (jej koszt). Zmiana feromonu następuje oczywiście tylko na znalezionej przez mrówke ścieżce.

Sposób parowania feromonu jest również bardzo prosty:

$$_{ij}(t) = (1 \qquad) \ _{ij}(t)$$

W powyższym wzorze współczynnik f określający szybkość parowania określamy eksperymentalnie.

3.2 Rezultaty doświadczeń

Algorytm AS był poddawany szeregom doświadczeń porównujących go z innymi typowymi heurystykami używanymi w problemie komiwojażera. Rezultaty były jednocześnie bardzo interesujące jak i troche niezadowalające. Z jednej strony algorytm AS dla względnie małych grafów był w stanie bardzo szybko znaleźć dosyć dobre rozwiązanie. Z drugiej jednak strony dla większych grafów nigdy nie zbliżył się on do uzyskanych innymi metodami najlepszych rozwiązań. Wskutek tych obserwacji algorytm AS zaczął ewoluować aż do powstania jego różnych wariantów, np. Max-Min AS, które wykazują się lepszymi cechami niż oryginalny AS.

4 Max-Min AS

Algorytm Max-Min AS jest modyfikacją oryginalnego algorytmu AS. Modyfikacje te sprowadzają się do 3 zmian:

- ilość feromonu na krawędziach ograniczana jest do przedziału $[\ _{min}, \ _{max}]$, służy to przeciwdziałaniu stagnacji, która była jednym z głównym powodów dla których AS nie spisywał się najlepiej dla większych grafów.
- początkowo na krawędziach obecnych jest $_{max}$ feromonu

- w każdej iteracji najlepsza droga ze znalezionych przez mrówki otrzymuje dodatkową ilość feromonu

Doświadczenia pokazały, że algorytmu typu Max-Min AS znajduje znacząco lepsze rozwiązania niż AS. Dodatkowo ważną jego zalęta jest też niewiele większe skomplikowanie algorytmu niż oryginalny AS.

5 Algoryty mrówkowe w praktyce

Po sukcesie w rozwiązywaniu problemu komiwojażera naturalne stały się próby aplikacji algorytmów mrówkowych do innych typowych problemów kombinatorycznych. Próbowano nimi rozwiązywać zarówno statyczne problemy kombinatoryczne (dane wejściowe nie zmieniające się w czasie) oraz dynamiczne (dane wejściowe ulegają zmianom w czasie). Do typowych statycznych problemów kombinatorycznych dla których istnieją bardzo dobre rozwiązania algorytmami mrówkowymi możemy zaliczyć:

- kolorowanie grafu
- przypisywanie fabryk do lokalizacji (quadratic assignment problem)
- szeregowanie zadań na maszynach (job shop scheduling)
- wyznaczanie tras pojazdów (vehicle routing problem)
- najkrótszy wspólny nad-ciąg (shortest common supersequence)

Jeśli chodzi o problemy dynamiczne, to głównym polem zastosowania algorytmów mrówkowych jest routing w sieciach komunikacyjnych. Chodzi tutaj zarówno o routing połączeniowy (pakiety tego samego połączenia przesyłane są tą samą drogą) jak i bezpołączeniowy (każdy pakiet może być przesyłany inną drogą).

6 Implementacje na maszynach równoległych

Bardzo ważnych zagadanieniem we współczesnym świecie jest możliwość implementacji algorytmów mrówkowych na maszynach równoległych. Z jednej strony można to zrobić bardzo prosto - przypisać każdą mrówkę do osobnego jednostki wykonawczej (procesora). Niestety takie podejście mimo wielu zalet (m.in. prostota implementacji) ma poważną wadę. Po pierwsze zauważmy, że z definicji ilość obliczeń przypadająca na pojedyńczą mrówkę jest względnie małą, natomiast ilość komunikacji z innymi mrówkami może być znacząca (musimy tu uwzględnić m.in. informacje o feromonach). Wskutek tego narzuty komunikacyjne okazują się zbyt duże w porównaniu z zyskami ze zrównoleglenia.

Najprościej rozwiązać ten problem poprzez zmniejszenie ziarnistości. Dokonuje się tego dzieląc daną kolonię mrówek na pewną ilość podkoloni, rozmieszczonych na róznych procesorach. Każda podkolonia jest symulowana osobno z

uwzględnieniem dodatkowych danych (np. przepływ feromonu między koloniami). Dzięki temu narzuty komunikacyjne stają się względnie małe w porównaniu do ilości obliczeń przeprowadzanej przez pojedyńcze procesory. Doświadczenia pokazują, że jest to bardzo skalowalne rozwiązanie a przyspieszenie uzyskiwane w ten sposób jest praktycznie liniowe ze względu na ilość procesorów.