Algorytmy Mrówkowe

Daniel Błaszkiewicz

Instytut Informatyki Uniwersytetu Wrocławskiego

11 maja 2011

Algorytmy mrówkowe to stosunkowo nowy gatunek algorytmów optymalizacyjnych stworzony przez Marco Dorigo w 1992 na podstawie obserwacji zachowań mrówek jako jednostek i działań mrowiska jako całości.

Powstały one na podstawie obserwacji zachowań mrowisk i służą do rozwiązywania problemów kombinatorycznych i optymalizacji problemów przedstawionych na grafach.

Typowy problem dla mrowiska - znalezienie źródła pożywienia, a następnie przetransportowanie go z powrotem do kolonii. Do dyspozycji ma tylko mrówki, będące agentami którzy:

- Są bardzo ograniczeni w percepcji
- Są indywidualnie zdolni tylko do bardzo prostych zachowań

Jak zaobserwowano mimo tych ograniczeń zdobywanie pożywienia odbywa się bardzo wydajnie. Jak to się dzieje?

Mrówki wydzielają substancje chemiczne zwane feromonami. Feromony pozostawiane są na trasie mrówek, umożliwiając w ten sposób komunikację. Feromony odparowują z czasem. Umożliwia to następujące rozwiązanie tego problemu:

- Wypuszczani są w miarę losowo poruszających się zwiadowcy
- Kiedy zwiadowca natrafi na źródło żywności zaczyna wracać mniej więcej w kierunku kolonii, znacząc trasę feromonem
- Mrówki w kolonii są przyciągane do feromonu, starając się iść mniej więcej w kierunku przez niego wskazywanym, znacząc swoje przejście feromonami
- Na wydajniejszych trasach w jednostce czasu chodzić będzie więcej mrówek, więc będą bardziej oznaczone, przyciągając coraz więcej mrówek
- Na mniej wydajnych trasach będzie chodziło coraz mniej mrówek, a feromon będzie odparowywał, czyniąc je jeszcze mniej atrakcyjnymi

Po pewnym czasie mrówki będą chodziły prawie wyłącznie po bardzo optymalnej trasie. Nie wymagało to od nich żadnego indywidualnego myślenia, ten system sam się organizuje. Te obserwacje doprowadziły do stworzenia algorytmów mrówkowych (ACO - Ant Colony Optimization).

Algorytmy mrówkowe są algorytmami o następujących cechach:

- optymalizacja prowadzona jest iteracyjnie przez pewną liczbę agentów
- agenci komunikują się poprzez informację zostawianą w otoczeniu
- agenci losowo przeszukują przestrzeń, faworyzując rozwiązania wskazywane im przez informację z otoczenia
- agenci mogą mieć własną pamięć, ale raczej nie posiadają rozbudowanej inteligencji

Jako że ACO jest stosunkowo nowym wynalazkiem definicja mówiąca co jest, a co nie jest algorytmem mrówkowym wciąż jest nieco płynna, ale generalnie te cztery wymienione cechy uważa się za czynnik decydujący.

ACO dają się zastosować do wszelkiego rodzaju problemów dających się sprowadzić do wyboru kolejności odwiedzania wierzchołków na grafie. Najbardziej typowym przykładem jest tutaj TSP, ale rozwiązują one również problem plecakowy, problem harmonogramowania produkcji czy problem kolorowania wierzchołków grafu.

Typowe rozwiązanie problemu algorytmem mrówkowych wygląda następująco:

1. Stwórz reprezentację zadania w postaci zadania do wykonania na grafie
2. Ustal początkową ilość feromonów na krawędziach
3. Ustal dodatkowe reguły za pomocą których agenci będą się poruszać wewnątrz grafu (zwykle jest to po prostu pamiętanie tablicy tabu już odwiedzonych wierzchołków, a także wybór pomocniczej heurystyki)
4. Zdecyduj w jaki sposób ruch agentów będzie wpływać na poziom feromonów na krawędziach grafu, a także jak oni sami będą się posiłkować już położonymi feromonami
5. Iteracyjnie wpuszczaj nowe grupy agentów (po usunięciu starej grupy) i pozwalaj im przemieszczać się pomiędzy wierzchołkami grafu zgodnie ze stworzonymi regułami
6. Po każdej iteracji pamiętaj najlepsze dotychczas znalezione rozwiązanie

Zastosujmy ten schemat do rozwiązania klasycznego problemu informatyki - problemu komiwojażera. Komiwojażer ma odwiedzić wszystkie miasta idąc jak najkrótszą trasą i wrócić do punktu startowego. Wybór punktu startowego nie ma tak naprawdę znaczenia, jako że wygenerowana ścieżka będzie cyklem przechodzącym przez wszystkie wierzchołki.

Jak zastosować tutaj ACO?

- Na każdą krawędź nałóż pewną stałą początkową ilość feromonów
- Każda mrówka ma pamiętać odwiedzone miasta i już do nich nie wracać
- Mrówki zaczynają swój marsz w losowo wybranych miastach
- Mrówki poruszają się po grafie wybierając metodą ruletki kolejny wierzchołek do którego przejdą kierując się ilością feromonów na krawędziach prowadzących do dozwolonych wierzchołków połączonych połączonych z tym w których się znajdują
- Kiedy wszystkie mrówki dojdą do końca następuje aktualizacja ilości feromonów. Na każdej krawędzi ilość feromonu jest mnożona przez ustalony na początku czynnik z przedziału $[0, 1]$, a następnie do odwiedzonych przez mrówki krawędzi dodawane są wartości odwrotnie proporcjonalne do długości ich rozwiązań
- Po zakończeniu aktualizacji ilości feromonów pamięci mrówek są usuwane i mrówki zaczynają maszerować na nowo

Najbardziej interesująca część to wybór kolejnego miasta. O ile mrówka mogłaby się kierować po prostu medodą ruletki po feromonach na krawędziach pomiędzy danym punktem a dopuszczalnymi, to pomija to analogię z rzeczywistymi mrówkami starającymi się przy słabej ilości feromonowej iść w "mniej-więcej"dobrym kierunku. Jest to dobre miejsce więc na zastosowanie funkcji heurystycznej mającej to zapewnić. Dodatkowo wprowadźmy parametry α i β pozwalające nam regulować znaczenie feromonów i heurystyki przy przeszukiwaniu.

Niech $h(i,j) = (odleglosc(i,j))^{-1}$, zaś $f(i,j)$ to ilość feromonu na krawędzi (i,j).

Wartość danej krawędzi w ruletce to $w(i,j) = f(i,j)^{\alpha} \cdot h(i,j)^{\beta}$

Szansa na to że dozwolona krawędź (i,j) zostanie wylosowana wynosi $\frac{w(i,j)}{\Sigma\{w(i,x)|\neg odwiedzone(x)\}}$

Dostajemy tutaj więc współczynniki:

- Ilość mrówek
- Współczynnik parowania
- Minimalną wartość feromonu na krawędzi
- Alfa (waga feromonu w wyborze)
- Beta (waga heurystyki w wyborze)

Dobiera się je eksperymentalnie.

Algorytmy mrówkowe dają oczywiście się zastosować również do problemów nie będących bezpośrednio zdefiniowanymi na grafie, ale dającymi się do takich sprowadzić. Przykład - problem plecakowy.

- Przedmiotom są przypisane wierzchołki w grafie pełnym
- Mrówki startują w dodatkowym wierzchołku startowym
- Kiedy mrówka wchodzi do wierzchołka to przypisany mu przedmiot jest wkładany do plecaka - o ile się mieści
- Po przejściu wszystkich mrówek dokładana jest ilość feromonu proporcjonalna do wartości włożonych do plecaka przedmiotów
- Cały czas jest pamiętana trasa która dała najlepsze rozwiązanie

Dzięki przerobieniu problemu plecakowego na problem znalezienia optymalnej permutacji pozwoliło na zastosowanie ACO na grafie pełnym. Zastosowanie dla innych problemów na takich grafach w sposób oczywisty analogiczne.

Nieco innym problemem od dwóch poprzednich jest problem kolorowania grafu. Polega on na takim pokolorowaniu wierzchołków grafu, aby żadne dwa sąsiednie nie były pokolorowane tak samo i żeby użyć możliwie najmniej kolorów. Graf nie musi być pełny - gdyby był rozwiązanie byłoby oczywiste. Jak tutaj zastosować więc ACO? Skąd powinny startować mrówki, i jak powinny chodzić aby odbyło się to wydajnie?

Rozwiązaniem jest nie wpuszczanie mrówek na kolorowany graf, ale na pełny graf pomocniczy, stosując przy tym takie reguły:

- Kiedy mrówka wchodzi do wierzchołka, przypisuje mu najmniejszą liczbę naturalną jakiej nie ma żaden z jego sąsiadów w kolorowanym grafie
- Mrówki mogą odwiedzać wierzchołki w dowolnej kolejności, niezależnie od ich faktycznego połączenia w kolorowanym grafie
- Jako że chcemy zminimalizować ilość użytych kolorów, po przejściu wszystkich mrówek dodawana ilość feromonu jest odwrotnie proporcjonalna do największej użytej przez daną mrówkę liczby naturalnej

Interpretacja jest oczywiście taka, że liczby naturalne przypisywane przez mrówki odpowiadają poszczególnym użytym kolorom. Punkt startowy z oczywistych powodów nie ma znaczenia.

Dotychczas pokazywane problemy polegały na sprowadzeniu ich do problemu znalezienia optymalnej permutacji, który następnie był w prosty sposób rozwiązywany na grafie. Czasami przydatne jest wprowadzenie także innej adaptacji, jak można zobaczyć na przykładzie problemu harmonogramowania produkcji.

Dane są:

- Maszyny wykonujące kolejne etapy produkcji, dla każdego etapu jedna, z których każda może pracować tylko nad jednym zadaniem na raz
- Zadania muszące przebyć wszystkie etapy produkcji, wszystkie w tej samej kolejności, choć mogące wymagać różnej ilości czasu na każdym etapie

Zadania trafiają do maszyn w ten sposób, że jeśli zadanie X odwiedziło maszynę n przed zadaniem Y, to również mają odwiedzić maszynę $n+1$ w tej samej kolejności.

Treścią problemu jest znalezienie takiego uszeregowania zadań aby czas do wykonania ostatniego etapu dla ostatniego z nich był jak najkrótszy.

Adaptacja do ACO wydaje się oczywista, jako że w zadaniu dosłownie chodzi o znalezienie optymalnej kolejności wykonania tych zadań. I faktycznie, pierwszy krok to stworzenie grafu z wierzchołkami odpowiadającymi poszczególnym zadaniom. W tym momencie można wypuścić mrówki w standardowy sposób podobnie jak w problemie plecakowym, ale okazuje się że lepsze efekty osiągane są jeśli zmieni się sposób wyboru kolejnego osobnika. W prostej metodzie ruletki jeśli mrówka będzie szła dobrą trasą, a następnie losowo z niej zejdzie, straci większość informacji co do tego w jakiej kolejności warto układać dalsze zadania.

Rozwiązanie - zamiast używać ruletki kierując się tylko feromonami na krawędziach wychodzących z obecnego wierzchołka lepiej kierować się sumą feromonów na krawędziach z już odwiedzonych wierzchołków. Przykład:

- Wierzchołki ponumerowane od 0-5, gdzie 0 to wierzchołek startowy nie odpowiadający żadnemu zadaniu.
- Odwiedzone wierzchołki $\{0,1,3,5\}$, nieodwiedzone $\{2,4\}$
- Niezależnie od wierzchołka w którym mrówka obecnie się znajduje (niech będzie to x), waga na ruletce dla przejścia do 2 wynosi $f(0,2) + f(1,2) + f(3,2) + f(5,2)$, zaś do 4 $f(0,4) + (1,4) + f(3,4) + f(5,4)$, zamiast po prostu odpowiednio $f(x,2)$ i $f(x,4)$.

Takie rozwiązanie pozwala mrówkom wciąż wykorzystywać wiedzę o względnej wartości położenia zadań w permutacji nawet jeśli zejdą one z najlepszej trasy. Eksperymentalnie stwierdzono iż sprawdza się ono lepiej, szybciej prowadząc do lepszych wyników.

Pośród innych problemów rozwiązywanych przez algorytmy mrówkowe znajdują się między innymi:

- kwadratowy problem przydziału (QAP)
- problem transportowy
- sortowanie sekwencyjne
- routing pakietów w sieciach

Jak widać na przykładzie harmonogramowania algorytmy mrówkowe można i czasem warto modyfikować względem ich podstawowego schematu. Jednym z miejsc gdzie można to zrobić jest wybór następnika. Jednym z podejść jest pokazane właśnie inne przekształcanie feromonów na szanse wyboru kolejnego wierzchołka do odwiedzenia. Poza prostym sumowaniem można również na przykład wziąć różne współczynniki α i β dla promowanie różnych sposobów przeszukiwania przestrzeni rozwiązań ($x^{0.5}$ promowałaby szerokie przeszukiwanie przestrzeni rozwiązań, x^2 eksploatację). Dalsze różnice będą w wyborze funkcji heurystycznych. Dla TSP była to po prostu odwrotność odległości, dla problemu plecakowego można zastosować funkcję promującą wybór przedmiotów o wysokim współczynniku $\frac{cena}{waga}$. Konkretne ulepszenia zależeć będą oczywiście od natury problemu.

Jak łatwo zauważyć mrówki wykonujące przedstawione rozwiązania problemów zachowują się inaczej niż te prawdziwe jeśli chodzi o zostawianie feromonu - zamiast robić to na bierząco, czekają nie dość aż same skończą swój marsz to jeszcze aż zrobią to pozostałe mrówki w tej iteracji. Generalnie po obserwacjach prawdziwych mrówek pojawiły się trzy pomysły na zachowanie tych wirtualnych:

1. podczas marszu mrówki zostawiałyby stałą ilość feromonu
2. podczas marszu mrówki zostawiałyby ilość feromonu proporcjonalną (lub odwrotnie proporcjonalną) do długości krawędzi po której idą
3. mrówki zostawiałyby feromon, w ustalony dalej sposób, dopiero po zakończeniu marszu

Ponieważ nie stosujemy tutaj prawdziwych mrówek, a jedynie wirtualne mrówki-agentów symulujących ich zachowanie, możemy dalej zdecydować które z nich i jak będą w ogóle zostawiać swój feromon. I znowu, trzy możliwości:

- Wszystkie mrówki zostawiają swoje feromony w tej samej ilości
- Wszystkie mrówki zostawiają swoje feromony, ale mrówka która stworzyła najlepsze rozwiązanie zostawia ich więcej lub każda mrówka zostawia ilość feromonu zależną od jakości jej rozwiązania
- Tylko najlepsza mrówka zostawia feromon

Interesujące są tutaj rozwiązanie drugie i trzecie. Rozwiązanie drugie jest standardowym, rozwiązanie trzecie (zwane strategią elitarną) prowadzi do większej zbieżności. Dobór strategii jest zależny od problemu - jeśli można się spodziewać sporej ilości optimów lokalnych strategia elitarna nie jest dobrym pomysłem. Jeśli mało, to pozwala ona dojść do rozwiązań stosunkowo szybko.

Parowanie feromonów również nie musi wyglądać zawsze tak samo. Poza oczywistymi rzeczami do modyfikacji takimi jak współczynnik globalnego parowania czy minimalna wartość feromonu jaka jest dopuszczalna, niektóre algorytmy wprowadzają dodatkowo parowanie lokalne. Działa ono w ten sposób że krawędź po której właśnie przeszła mrówka traci pewien procent swojego feromonu niezależnie od późniejszego parowania globalnego. Służy to ku skłonieniu mrówek do szerszej eksploracji i jest czasem warte eksperymentalnego sprawdzenia co do skuteczności.

Czy ACO to rodzaj algorytmów ewolucyjnych? Na pewno wykazują do nich pewne podobieństwa. Przykładowo dla problemu rozwiązywanego na statycznym grafie, takim jak TSP, możnaby próbować zdefiniować następujący algorytm ewolucyjny:

- Ewolucja toczy się na grafie TSP
- Populacja składa się z jednego osobnika, na którego genotyp składa się wewnętrzna reprezentacja grafu rozkładu feromonów i dotychczas najlepszej znalezionej ścieżki jaka jest mu znana
- W każdej iteracji ewolucji przepuszczamy wirtualne mrówki po grafie TSP, kierując nimi za pomocą grafu rozkładu feromonów z osobnika i na tej podstawie zamieniamy osobnika na nowego z nowym grafem rozkładu feromonów (oczywiście bazując na poprzednim) i ew. nową znalezioną najlepszą trasą
- Jeśli chcemy odczytać fenotyp osobnika patrzymy tylko na jego najlepszą znalezioną trasę

Jak widać przy sporej dozie dobrej woli da się wyrazić ACO w języku bardzo bliskim algorytmom ewolucyjnym. Są to bardzo podobne, choć jednak różne dziedziny algorytmów, i to który należy zastosować do jakiego problemu najlepiej rozpatrywać indywidualnie.

KONIEC