Algorytmy Mrówkowe

Daniel Błaszkiewicz

11 maja 2011

1 Wprowadzenie

Popularnym ostatnimi laty podejściem do tworzenia nowych klas algorytmów do szukania rozwiązań problemów nie mających algorytmów rozwiązujących je w czasie wielomianowym jest szukanie inspiracji w naturze. Procesy ewolucji i naturalnej selekcji promowały pojawianie się jak najbardziej wydajnych rozwiązań - jeśli udałoby się zaadaptować je do użytku przy rozwiązywaniu problemów informatycznych może się okazać że sprawdzają się one bardzo dobrze także i w tym nowym kontekście. Tak powstały na przykład zainspirowane samym procesem ewolucji algorytmy ewolucyjne, a także omawiane przeze mnie algorytmy mrówkowe.

Algorytmy mrówkowe, znane pod angielską nazwą ACO (Ant Colony Optimization) są stosunkowo niedawnym wynalazkiem. Zaproponowane zostały one w pracy doktorskiej Marco Dorigo w roku 1991 (opublikowanej w roku 1992), zainspirowanej opublikowaną w 1989 roku pracy o zachowaniu argentyńskich mrówek. Od tego czasu schemat tych algorytmów został rozwinięty do swojej obecnej postaci, którą przedstawię dalej w tym opracowaniu.

2 Algorytmy

2.1 Mrówki w naturze

Jak więc działają algorytmy mrówkowe, i skąd one się wzięły? Przyjrzyjmy się zadaniu stojącemu przed typową kolonią mrówek. Kolonia jest zamieszkała przez bardzo małe owady, indywidualnie niezdolne do prowadzenia zbyt skomplikowanych rozumowań, poruszające się głównie na bazie prostych instynktów. Mrówki muszą jednak zdobywać pożywienie, które może znajdować się bardzo daleko - a muszą to zrobić w sposób zorganizowany. I rzeczywiście: mrówki znajdują pożywienie, wytyczają dobre ścieżki i nimi transportują je spowrotem do kolonii. Jak więc to robią, skoro nie ma wśród nich żadnej która by kierowała pracą?

Oferowane wyjaśnienie, na którym oparta jest również idea algorytmów mrówkowych, opiera się na feromonach. Wygląda ono w następujący sposób:

- Zakłada się że mrówki mają pewne poczucie kierunków
- Mrowisko wypuszcza poruszających się mniej więcej losowo zwiadowców
- Kiedy zwiadowca natrafi na coś interesującego wraca do kolonii, znacząc swoją trasę feromonami
- Feromony przyciągają więcej mrówek, które zaczynają się kierować mniej więcej wyznaczoną przez nie trasą
- Kiedy te mrówki docierają do pożywienia, zaczynają nieść je spowrotem, znów znacząc swoją ścieżkę feromonami
- W ten sposób wyznaczana jest feromonowa trasa, która przyciąga coraz więcej mrówek i pozwala im na sprawny transport pożywienia spowrotem do kolonii

Skąd jednak ten aspekt optymalnych ścieżek? Mrówki przyciągane są do feromonów, ale nie muszą iść dokładnie po feromonowej ścieżce - zwłaszcza początkowo kiedy jest ona bardzo słaba. Jeśli mrówki miałyby do wyboru dwie trasy, obie tak samo pociągające od strony leżących na nich feromonów, mrówki wybierałyby pomiędzy nimi losowo. Na obie trasy trafiałaby taka sama liczba mrówek, więc oczywiście natężenie ruchu byłoby większe na krótszej z tras. Stąd z czasem nagromadziłaby ona więcej feromonu, więc stałaby się trasą preferowaną. Dodatkowo, feromony parują z czasem, co coraz bardziej zwiększałoby proporcjonalną różnicę w ich natężeniu pomiędzy obiema trasami, sprawiając że krótsza trasa stawałaby się jeszcze bardziej popularna, podczas gdy dłuższa coraz bardziej byłaby porzucana - aż ostatecznie zostałaby porzucona całkowicie. I tak oto mrówki samoczynnie organizują się i wybierają dobre trasy.

2.2 Mrówki w informatyce

Jak jednak zastosować te obserwacje do stworzenia algorytmów? Przyjrzyjmy się jakie są tutaj komponenty:

- Agenci o prostej inteligencji komunikujący się ze sobą tylko poprzez poziom feromonów zostawianych w otoczeniu
- Otoczenie oferujące ścieżki o różnych sumarycznych długościach
- Feromony rozłożone w otoczeniu sterujące ruchami agentów
- Znajdowane przez agentów ścieżki przekładają się na proponowane przez nich rozwiązania

Oczywistym otoczeniem do zastosowania algorytmów mrówkowych są grafy. Agenci poruszaliby się po nich, pamiętając swoje ścieżki, zostawiając feromony i generując coraz lepsze rozwiązania. Mrówki będą poruszać się iteracyjnie po grafie, więc należy zdecydować ile mrówek będzie przechodzić graf w danej iteracji. Dalej należy ustalić co mrówki będą pamiętać i jak będzie to wpływać na ich opcje w poruszaniu się. Nawet jeśli konstruujemy całkowicie standardowy algorym mrówkowy, aby pójść dalej w tym miejscu musimy podjąć kilka dużych decyzji.

Pierwszą z nich jest wybranie sposobu w jaki mrówki będą wybierać kolejne segmenty swoich ścieżek. Typowo po dojściu do danego wierzchołka mrówka wybierać będzie spomiędzy dozwolonych dalszych ścieżek metodą ruletki. Wartości dla tej metody dla każdej ze ścieżek wyliczane są korzystając z czterech komponentów:

1. Wartości funkcji feromonu dla danego segmentu
2. Powiązanego z nią współczynnika α
3. Wartości pomocniczej funkcji heurystycznej dla danej ścieżki
4. Powiązanego z nią współczynnika β

Wartość funkcji feromonu to zwykle po prostu ilość feromonu na ścieżce, choć dla niektórych problemów (w tym na przykład opisywanego dalej problemu harmonogramowania produkcji) warto zastosować nieco inny sposób jej wyliczania. Ta wartość jest podnoszona do potęgi α, gdzie współczynnik α jest ustaloną na początku wartością mówiącą jak bardzo mrówki mają się kierować ilością feromonów. Wyższe wartości prowadzą do intensywniejszej eksploracji okolic już znalezionych ścieżek, niższe do eksploracji w poszukiwaniu nowych.

Naturalne mrówki mają pewne ogólne pojęcie co do kierunków i pewne dodatkowe instynkty kierujące ich zachowaniem. Symulowane są one tutaj poprzez zastosowanie odpowiedniej dla problemu funkcji heurystycznej, mającej promować rozwiązania które lokalnie wyglądają obiecująco. Podobnie jak w przypadku funkcji feromonu również i z wartością heurystyczną powiązana jest wartość do potęgi której jest ona podnoszona (współczynnik β), podobnie promując eksplorację lub eksploatację w zależności od swojej wartości.

Ostatecznie wartość przypisywaną dla danej ścieżki pomiędzy v i n w metodzie ruletki można zapisać więc w następujący sposób: $f(v,n)^\alpha \cdot h(v,n)^\beta$

Kolejną decyzją do podjęcia jest wybór sposobu zostawiania feromonów przez mrówki. Prawdziwe mrówki zostawiają je na bierząco podczas marszu, ale fakt że mamy tutaj do czynienia z mrówkami wirtualnymi pozwala nam odejść od tego i zastosować inne rozwiązania. Typowo stosowane są tutaj trzy podejścia:

1. Mrówki zostawiają feromony na bieżąco w stałych ilościach na każdej krawędzi po której przechodzą
2. Mrówki zostawiają feromony na bieżąco w ilościach odzwierciedlających lokalną jakość danej krawędzi (na przykład jeśli chcemy zminimalizować sumaryczną długość całej ścieżki, to zostawiana byłaby ilość odwrotnie proporcjonalna do długości wybranej krawędzi)

3. Mrówki wykonują swój marsz bez zostawiania feromonów, a następnie kładą je na ścieżce w sposób odzwierciedlający globalną jakość ścieżki

Wariant pierwszy jest prawie całkowicie zależny od zastosowania dobrej heurystyki - feromony stanowią w nim tylko swoiste wsparcie ścieżek na które kieruje mrówki heurystyka. Bez heurystyki znajdowane ścieżki byłby całkowicie losowe. Nie jest to prawda w wariancie drugim, gdzie lokalna jakość krawędzi ma odzwierciedlenie w rozkładanych feromonach i w ten sposób nadaje im większe znaczenie przy sterowaniu mrówkami.

Wariant trzeci jest typowo najlepszym i najczęściej stosowanym z tych trzech wariantów, rozkładając feromony wedle globalnej perspektywy (lokalnie bardzo dobra ścieżka może prowadzić do wierzchołków z których dalej ścieżki byłby bardzo mało wydajne). Gdyby taki globalny widok nie był dostępny warto jednak pamiętać o wariantach pierwszym i drugim.

Zastosowanie wariantu trzeciego prowadzi do możliwości kolejnej zmiany względem naturalnego zachowania mrówek. W naturze każda z mrówek zostawia swój feromon tak samo. Wirtualne mrówki zostawiające feromony dopiero po zakończeniu iteracji nie mają takiego ograniczenia. I tak, znów trzy możliwości:

1. Każda z mrówek zostawia feromony w tej samej ilości
2. Każda z mrówek zostawia feromony, lepsze rozwiązania zostawiają więcej feromonów
3. Tylko najlepsza z mrówek zostawia feromony

Wariant pierwszy jest oczywiście bardzo podobny do wariantu pierwszego z poprzedniego wyboru. Bardziej interesujące są warianty drugi i trzeci, odpowiednio promujące eksplorację i eksploatację. Wybór pomiędzy nimi zależy od specyfiki rozwiązywanego problemu.

Ostatnią rzeczą jest parowanie. Globalne parowanie jest typowo wykonywane poprzez mnożenie wszystkich wartości feromonów na krawędziach przez pewną stałą wartość z przedziału $(0, 1)$ po przejściu wszystkich mrówek, choć można oczywiście wymyślić również i inny schemat parowania. Dodatkowo stosuje się czasem mające niewiele wspólnego z naturą parowanie lokalne, gdzie po przejściu mrówki daną krawędzią wartość feromonu na niej jest zmniejszana (również typowo poprzez przemnożenie jej przez stałą z przedziału $(0, 1)$). Robi się to w celu promowania eksploracji kosztem eksploatacji. Warto jednak upewnić się że ilość feromonów nie zostanie odparowana poniżej pewnej wartości, gdyż od tego momentu mogłaby zginąć w błędach zaokrągleń i ścieżka nie byłaby sprawdzana już nigdy.

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

3 Przykłady zastosowań

3.1 TSP

Aby dobrze zrozumieć działanie algorytmów mrówkowych, najlepiej spojrzeć na ich działanie dla konkretnych problemów. Narzucającym się problemem do rozwiązywania przez ACO jest problem komiwojażera - jest to problem w którym należy wyznaczyć najkrótszą ścieżkę na grafie odwiedzającą wszystkie jego wierzchołki dokładnie raz. Zastosujmy więc mrówki zachowujące się jak komiwojażer i zobaczmy jak będą się one zachowywać. Tak więc:

- Mrówki będą pamiętać listę odwiedzonych przez siebie wierzchołków w kolejności ich odwiedzania. Zauważmy przy tym że punkt początkowy dla danej trasy nie ma znaczenia, jako że niezależnie od tego jaki zostanie wybrany generować będziemy cykl Hamiltona
- Wartości feromonów na krawędziach będą wpływać w sposób bezpośredni na wartość w metodzie ruletki
- Lokalna pomocnicza funkcja heurystyczna to po prostu odwrotność długości danej krawędzi
- Mrówki będą zostawiać feromony dopiero po przejściu całej trasy przez wszystkie z mrówek. Na każdej z odwiedzone przez nie krawędzi zostawiana będzie odwrotność długości całej przebytej przez nie ścieżki
- Współczynnik parowania, minimalna ilość feromonu na krawędzi, α, β i ilość mrówek początkowo zostaną dobrane na wyczucie, należy je poprawiać wraz z obserwacją algorytmu w działaniu

Sam algorytm działać będzie tak:

- Dla każdej mrówki ze zbioru przejdź całą trasę. Wykonaj to losując miasto początkowe, a następnie przechodząc do wciąż nieodwiedzonych miast wybierając kolejne miasta metodą ruletki i pamiętając przebytą dotychczas trasę. Kiedy wszystkie miasta zostaną odwiedzone załóż że następnym krokiem mrówki jest powrót do punktu początkowego
- Kiedy wszystkie z mrówek zakończą marsz dodaj na krawędziach przez które przeszły odwrotności długości znalezionych przez nie ścieżek
- Jeśli któraś z mrówek w tej iteracji przeszła trasą krótszą niż dotychczas najkrótsza znaleziona, zapamiętaj tą nową trasę jako nową najkrótszą znalezioną.
- Odparuj feromony na wszystkich krawędziach mnożąc je przez współczynnik parowania uważając aby nie zejść poniżej dopuszczalnego minimum feromonu na krawędzi
- Wykasuj pamięć dla każdej z mrówek i zacznij kolejną iterację

Podczas kolejnych iteracji przy dobrze dobranych współczynnikach mrówki znajdują w oczywisty sposób coraz lepsze i lepsze rozwiązania, chodząc losowo ale będąc coraz bardziej przyciąganymi do obiecujących ścieżek i coraz mniej do mniej obiecujących.

Ta właśnie metoda została zastosowana przy pisaniu przykładowego programu. Program jest napisany w C pod Linux'a i wymaga biblioteki SFML. Obsługuje się go poprzez zmiany w zdefiniowanych wartościach na początku kodu. Zmieniać można współczynniki algorytmu mrówkowego, ilość losowo wygenerowanych miast, ustalać seed funkcji pseudolosowej (lub przełączyć na stosowanie time(NULL)), a także modyfikować pewne opcje graficzne. Program rysuje miasta w ich miejscach położenia, najlepszą dotychczas znalezioną ścieżkę i rozkład feromonów (0% przezroczystości dla krawędzi o największej ilości feromonów, 100% dla krawędzi o najmniejszej, proporcjonalnie dla reszty) wykonując cały czas kolejne iteracje głównej pętli algorytmu. Jest to program którego działanie jest oczywiście bardzo oparte na dobrym doborze współczynników - dla dobrego doboru eksperymentalnie dość szybko widać zbieżność do dobrze wyglądających rozwiązań. Program jest przeznaczony do ogólnego przezentowania zasady działania algorytmów mrówkowych.

3.2 Problem plecakowy

Dla TSP algorytm mrówkowy daje się rozwiązać w sposób całkowicie naturalny, gdyż jest to od razu problem znaleziena dobrej ścieżki w grafie. Przyjrzyjmy się jednak problemowi plecakowemu, w którym mamy skończoną ilość elementów o różnych wagach i wartościach i plecak mogący zabrać tylko przedmioty do określonej sumarycznej wagi, a mamy odpowiedzieć jaka jest najwięcej warta kombinacja przedmiotów które można zabrać w tym plecaku. Nie ma tu nic o ścieżkach ani grafach, więc początkowo nie wygląda to na problem w którym można zastosować algorytmy mrówkowe.

Wszystko zależy jednak od sposobu reprezentacji rozwiązań. Zamiast pamiętać zbioru przedmiotów, pamiętajmy ich permutację. Dla danej permutacji próbujmy po kolei wkładać przedmioty do plecaka. Jeśli przedmiot się mieści, jest dokładany. Jeśli nie, jest pomijany. W szczególności jeśli na początku znajdowałyby się przedmioty ze zbioru optymalnych, to wszystkie oczywiście trafiłyby do środka a reszta byłaby pominięta - tak więc optymalne rozwiązanie jest pośród tak reprezentowanych rozwiązań.

Zmiana problemu z generowania zbioru do generowania permutacji pozwala jednak już w naturalny sposób przekształcić to na problem odwiedzania w odpowiedniej kolejności wszystkich wierzchołków w grafie - wystarczy stworzyć graf pełny o wierzchołkach etykietowanych wszystkimi elementami permutacji i znaleźć ścieżkę odwiedzającą wszystkie wierzchołki. Kolejność odwiedzania wierzchołków to kolejność pojawiania się ich w permutacji. Jak widać było w poprzednim przykładzie, jest to jak najbardziej rozwiązywalne przez algorytm mrówkowy. W przeciwieństwie jednak do niego wybór pierwszego elementu ma znaczenie. Aby także i tutaj móc wykorzystać własności algorytmu mrówkowego dodajmy sztuczny wierzchołek startowy z którego wypuszczane będą mrówki, również połączony ze wszystkimi wierzchołkami - w ten sposób już aby dojść do pierwszego elementu mrówka będzie musiała korzystać z krawędzi na grafie, a na tej będą leżały feromony. Ustalmy jeszcze pomocniczą heurystykę sprawiającą aby atrakcyjność danej ścieżki była wprost proporcjonalna do stosunku $\frac{cena}{waga}$ danego przemiotu. Zostawiane przez mróki feromony niech będą wprost proporcjonalne do wartości przedmiotów włożonych do plecaka w wygenerowanych przez nie rozwiązanich.

W ten sposób otrzymamy sytuację w której najlepszy wybór przedmiotów trafiających do plecaka jest równoznaczny z wyborem najlepszej ścieżki w grafie, a jak było widać na przykładzie TSP jest to zadanie rozwiązywalne algorytmami mrókowymi.

3.3 Kolorowanie grafu

W obu powyższych przypadkach problem rozwiązywany przez algorytm mrówkowy był problemem kombinatorycznym rozwiązywanym przez znajdowanie optymalnej ścieżki na grafie pełnym. Co jednak z problemami które od razu są zadane na grafie? Przykładem takiego problemu jest problem kolorowania grafu, w którym należy tak pokolorować wierzchołki w grafie aby użyć możliwie najmniej kolorów, a żadne dwa sąsiednie wierzchołki nie były pokolorowane tak samo.

Począkowy graf nie musi być grafem pełnym. Oznacza to że gdyby po prostu wpuścić na niego mrówki, najprawdopodobniej przechodziłyby po poszczególnych wierzchołkach wielokrotnie, czyniąc rozwiązanie mało wydajnym.

Rozwiązaniem jest nie wypuszczać mrówek bezpośrednio na ten graf, ale stworzyć graf pomocniczy o takim samym zbiorze wierzchołków jak graf oryginalny, ale pełnym zbiorze krawędzi. Ponumerujmy możliwe kolory i wypuśćmy na tym grafie mrówki z interpertacją że odwiedzany wierzchołek jest kolorowany na najniższy kolor nie użyty dotychczas do pokolorowania żadnego z sąsiadów. Zostawiana ilość feromonów byłaby odwrotnie proporcjonalna do ilości użytych różnych kolorów, podobnie heurystyka dawałaby wartości odwrotnie proporcjonalne do ilości użytych dotychczas różnych kolorów po danym ruchu. Uniezależnienie w ten sposób możliwych przejść mrówek od krawędzi w oryginalnym grafie oznacza że każdy wierzchołek będzie odwiedzany dokładnie raz.

3.4 Harmonogramowanie produkcji

Rozwiązania wszystkich trzech poprzednich problemów stosowały w metodzie ruletki fukcję biorącą po prostu wartość feromonu na krawędzi. Czasami warto jednak tą funkcję zmodyfikować.

Zadanie harmonogramowania produkcji wygląda w następujący sposób: mamy pewną ilość zadań które muszą zostać przeprowadzone przez linię produkcyjną. Wszystkie zadania muszą odwiedzić wszystkie maszyny na linii w tej samej kolejności - jeśli zadanie A trafiło na linię przed zadaniem B, to każda z maszyn linii musi przetworzyć zadanie A przed zadaniem B. Każde z zadań może wymagać różnych ilości czasu na różnych maszynach. Jaka kolejność przetwarzania zadań pozwala na najszybsze przetworzenie ich wszystkich?

Ponieważ jest to problem w którym należy wygenerować optymalną permutację można zastosować ten sam pomysł co przy problemie plecakowym - stwórzmy graf pełny z wierzchołkami odpowiadającymi kolejnym zadanion i "ślepym" wierzchołkiem startowym, a następnie wypuśćmy mrówki przechodzące przez wszystkie wierzchołki i zostawiające feromony w ilościach odwrotnie proporcjonalnych do sumarycznego czasu rozwiązania (przykładowa heurystyka - czas o który wydłuży się rozwiązanie poprzez dodanie danego zadania). Weźmy jednak sytuację w której mrówki stwierdziły że po wierzchołku A warto odwiedzić wierzchołek B - ta informacja będzie zapisana w feromonach. Co jednak jeśli po odwiedzeniu wierzchołka A mrówka losowo wejdzie do wierzchołka C? Umieszczenie wierzchołka B dalej będzie opłacalne (bo jest to dalej rzecz następna po wierzchołku A), ale przy prostym obserwowaniu feromonów ta informacja będzie od tego miejsca dla mrówki niedostępna. Mrówki dalej będą znajdować dobre rozwiązania, ale wydajność ucierpi.

Rozwiązaniem tego problemu jest zastosowanie innej metody obliczania funkcji feromonu. Zamiast patrzyć po prostu na ilości feromonów na krawędzi pomiędzy danym wierzchołkiem i jego odwiedzalnymi sąsiadami, patrzmy na sumę feromonów na wszystkich krawędziach do odwiedzalnych wierzchołków z wierzchołków już odwiedzonych. W ten sposób ta informacja zostanie zachowana, a wydajność w znajdowaniu dobrych permutacji wzrośnie.

4 Warianty i hybrydy

Jak widać w podstawowych algorytmach mrówkowych istotnymi modyfikowalnymi fragmentami są:

- Przekształcenie problemu na problem wytyczania ścieżki w grafie tak aby algorytm mrówkowy działał w nim wydajnie
- Dobór funkcji heurystycznej
- Dobór metody obliczania wpływu feromonów na ruch mrówek
- Dobór wartości α i β mówiącej o wypływie funkcji feromonów i funkcji heurystycznej przy metodzie ruletki
- Dobór metod i współczynników parowania
- Dobór ilości mrówek
- Dobór odpowiedniej metody zostawiania feromonów przez mrówki

Podstawowe warianty rozwiązań będą wprowadzać modyfikacje pośród powyższych ustalanych elementów. Algorytmy mrówkowe można jednak modyfikować inaczej - najbardziej popularną modyfikacją jest hybrydyzacja ich z algorytmami ewolucyjnymi, prowadząca do bardzo wydajnych rozwiązań. Na przykład. zmodyfikowany tak algorytm mrówkowy generuje jedne z najlepszych rozwiązań dla TSP. Przykład takiej modyfikacji to traktowanie mrówek jako populacji osobników modyfikowanych ewolucyjnie co pełną iterację - osobniki mogłyby przechowywać własne współczynniki α i β, a następnie ewoluować w zależności od jakości znajdowanych rozwiązań. Już ta

prosta modyfikacja pozwala na samodostosowywanie się tych dwóch wartości na bierząco, rozwiązując problemy wywołane przez dobrenie nieoptymalnych wartości na początku i możliwe że znacząco usprawniając działanie algorytmu.

Algorytmy mrówkowe jako takie są raczej schematem rozwiązywania problemów - dla konkretnych problemów warto czasami zastanowić się nad modyfikacjami i odejściem od schematu. Stanowią one jednak ciekawe nowe podejście do problemu szukania rozwiązań dla problemów interpretowalnych jako problem znajdowania optymalnej ścieżki na grafie (najlepiej pełnym) i ich zastosowanie jest często warte rozważenia.

Literatura