Problem komiwojażera dla algorytmów mrówkowych

Patryk Majewski
250134

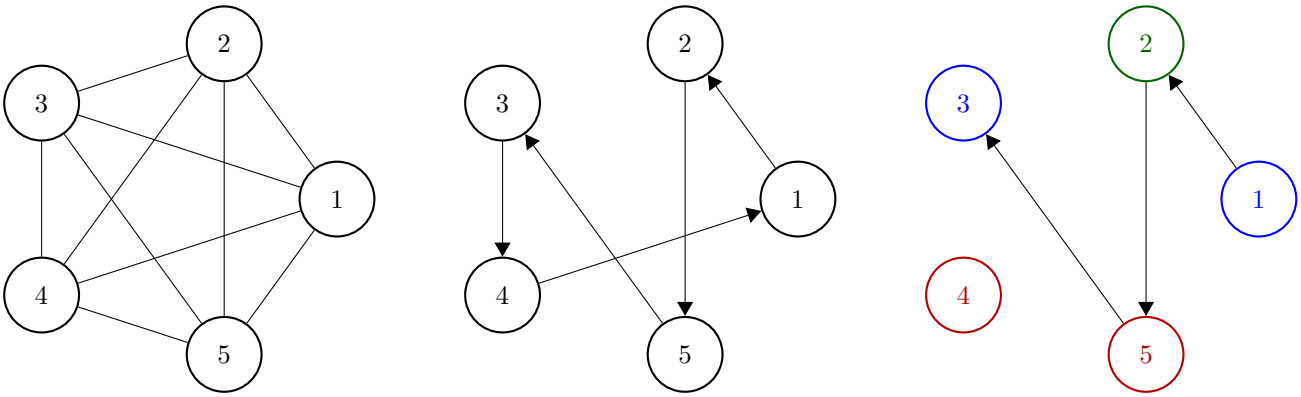
1 Rozważany problem

Problem komiwojażera (Travelling salesperson problem, TSP) jest popularnym zagadnieniem z zakresu optymalizacji dyskretnej. Mając zadany układ miast wraz z odległościami między nimi, sprzedawca musi wykonać kurs, w którym odwiedzi każde z nich jak najniższym kosztem. Prawidłowa trasa kończy się i zaczyna w tym samym mieście, a każde z pozostałych pojawia się na niej dokładnie raz. W tradycyjnej odmianie zadania mamy zatem nieskierowany graf pełny, w którym wierzchołki reprezentują miasta, a krawędzie o odpowiednich wagach – połączenia i ich długości. Poszukiwaną odpowiedzią jest cykl Hamiltona, w którym suma wag przebytych krawędzi jest możliwie najmniejsza.

Jednym z rozszerzeń podstawowej wersji problemu, opisanym w [4], jest uogólniony TSP (GTSP). Wierzchołki grafu wejściowego reprezentujące miasta są w nim zgrupowane w pewną liczbę (niekoniecznie rozłącznych) niepustych klastrów $V_1, V_2, ..., V_n$, takich że

$$\bigcup_{i=1}^{n} V_i = V,$$

gdzie V jest zbiorem wszystkich wierzchołków grafu. Zadaniem handlarza jest tym razem odwiedzenie każdego z klastrów przynajmniej raz, wracając do tego, z którego rozpoczął podróż (ale niekoniecznie do tego samego miasta). Wariant rozważany w pracy [4], zwany E-GTSP, zakłada, że każdy klaster z wyjątkiem startowego zostanie odwiedzony dokładnie raz. Warto zauważyć, że tradycyjna definicja problemu jest szczególnym przypadkiem E-GTSP, w którym każdy wierzchołek jest jedynym elementem swojego klastra.



Rysunek 1: Pierwszy graf prezentuje pełny układ pięciu miast i połączeń między nimi. Na grafie drugim umieszczone zostało przykładowe rozwiązanie dla tradycyjnego wariantu TSP – kończymy w tym samym mieście, z którego wyruszyliśmy. Trzeci graf przedstawia przykładowe rozwiązanie dla E-GTSP – miasta podzielono na klastry (oznaczone kolorami), kończymy w tym samym klastrze, z którego zaczęliśmy, ale niekoniecznie w tym samym mieście (chociaż oczywiście i takie rozwiązanie byłoby poprawne).

2 Zastosowana heurystyka

Algorytmy mrówkowe (Ant colony optimization, ACO) stanowią populacyjne podejście optymalizacyjne wzorowane na zachowaniach zaobserwowanych u mrówek. Idea opiera się na sposobie, w jaki owady odnajdują najkrótszą ścieżkę prowadzącą do pożywienia bez użycia wzroku [4]. Każda mrówka pozostawia na swojej ścieżce określoną ilość feromonu, który zachęca jej następców do podążenia tą samą trasą. Krótsza droga zostanie przebyta szybciej, uzyska zatem przewagę w intensywności feromonu, co z kolei spowoduje wybranie jej przez kolejne mrówki, które zwiększą tę przewagę jeszcze bardziej. W efekcie cała kolonia owadów dąży do poruszania się tą samą, najkrótszą ścieżką.

Faktyczna procedura optymalizacyjna składa się zasadniczo z trzech kroków: konstrukcji rozwiązań, opcjonalnego przeszukiwania lokalnego oraz aktualizacji poziomów feromonu. Podstawowymi parametrami algorytmu są liczebność populacji i początkowe rozmieszczenie feromonu. Liczba mrówek jest parametrem, który stanowi nieoczywisty kompromis – większa populacja pozwala nam na dokładniejszą eksplorację przestrzeni, natomiast zwiększa czas trwania jednej iteracji, może zatem spowodować, że algorytm nie będzie wystarczająco ekspansywny. Jest to problem tym bardziej, że powiększenie populacji samo w sobie nie przyspieszy stabilizacji algorymu pod względem doboru rozwiązania.

Faza konstruowania przedstawia podróż pojedynczej mrówki, która będąc w punkcie i, wybiera j ze zbioru nieodwiedzonych jeszcze miejsc na podstawie dwóch czynników: informacji heurystycznej ($\eta_{i,j}$) i poziomu feromonu ($\tau_{i,j}$) [2]. Przez informację heurystyczną rozumie się wpływ podjętej decyzji na wynik ścieżki – jest on niezmienny w czasie trwania algorytmu. Wprowadzenie η powoduje, że w odróżnieniu od prawdziwych mrówek, agenci konstruujący ścieżki nie ignorują całkowicie "bodźców wizualnych" [4]. Będąc w punkcie i, wybór celu j ze zbioru Ω nieodwiedzonych jeszcze miejsc następuje z prawdopodobieństwem

$$p_{i,j} = \frac{\tau_{i,j}^{\alpha}\eta_{i,j}^{\beta}}{\sum_{h\in\Omega}\tau_{i,h}^{\alpha}\eta_{i,h}^{\beta}},$$

gdzie $\eta_{i,j} = 1/d_{i,j}$, $d_{i,j}$ jest kosztem przejścia z i do j, a α i β są parametrami pozwalającymi na kontrolę wpływu obu czynników na decyzje mrówek. Zbyt duże α może spowodować utknięcie w lokalnym minimum i znacznie ograniczyć okazje do poprawy błędów, natomiast wzrost β zbliża algorytm do podejścia zachłannego, w którym wybierany jest zazwyczaj najkrótszy dostępny odcinek. Po odwiedzeniu wszystkich miejsc mrówka domyka cykl, wracając do położenia początkowego.

Wykorzystanie metod przeszukiwania lokalnego (jak na przykład 2-opt) na skonstruowanych rozwiązaniach ma z reguły pozytywny wpływ na wyniki działania algorytmu [2].

Poziomy feromonu τ zmieniają się pod wpływem tras obranych przez mrówki. Aby zasymulować zjawisko czasowej przewagi krótszych ścieżek, przyrost feromonu jest odwrotnie proporcjonalny do ich długości, zatem dla k-tej mrówki

$$\Delta\tau_{i,j}^{k} = \frac{Q}{L^{k}},$$

gdzie Q jest parametrem, a L^k całkowitą długością trasy przebytej przez k-tą mrówkę [4]. Wartość Q powiązana jest z początkowym poziomem feromonu. Biorąc pod uwagę wagi połączeń, warto ustalić te parametry w taki sposób, żeby pojedyncza iteracja nie powodowała ogromnej (ani też skrajnie małej) zmiany w poziomie feromonu na pojedynczej krawędzi.

Celem zapobiegnięcia nieskończonej akumulacji feromonu, wprowadza się schematy ulatniania [2]. W najprostszym przypadku jest to po prostu parametr $\rho \in [0, 1]$, że w momencie $t + 1$ mamy

$$\tau_{i,j}(t+1) = \rho \cdot \tau_{i,j}(t) + \sum_{k} \Delta\tau_{i,j}^{k},$$

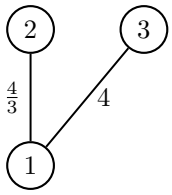
przy czym ρ bliskie zeru zbliża algorytm do losowego przeszukiwania, natomiast zbyt duże może utrudniać eksplorację przestrzeni.

Z uwagi na podobieństwo modeli, na których opierają się TSP i ACO, tradycyjna implementacja algorytmu mrówkowego rozwiązująca ten problem jest dosyć naturalna.

Celem zaprezentowania idei algorytmu, przyjrzymy się fragmentowi przestrzeni dla parametrów τ_0, α, β oraz Q równych 1 oraz $\rho = 0.5$. Załóżmy, że dwie mrówki startują z punktu 1, a następnie okazuje się, że każda z nich wybiera inną trasę. Mrówka, która wybrała krótszą trasę, uwalnia na niej więcej feromonu.

$$p_{1,2} = \frac{0.75}{0.75 + 0.25} = 0.75 \qquad \Delta\tau_{1,2} = 0.75 \qquad \tau_{1,2} = 0.5 \cdot 1 + 0.75 = 1.25$$

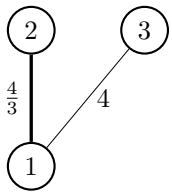
$$p_{1,3} = \frac{0.25}{0.75 + 0.25} = 0.25 \qquad \Delta\tau_{1,3} = 0.25 \qquad \tau_{1,3} = 0.5 \cdot 1 + 0.25 = 0.75$$



W kolejnej iteracji mrówki podejmują decyzję na podstawie zmienionych już poziomów feromonu – reprezentowanych na obrazku poprzez grubość linii. Możemy zauważyć, jak zmieniło się prawdopodobieństwo wyboru obu miast:

$$p_{1,2} = \frac{1.25 \cdot 0.75}{1.25 \cdot 0.75 + 0.75 \cdot 0.25} \approx 0.83$$

$$p_{1,3} = \frac{0.75 \cdot 0.25}{1.25 \cdot 0.75 + 0.75 \cdot 0.25} \approx 0.17$$



Oczywiście w faktycznej implementacji zmiana feromonu następuje na podstawie długości całej trasy (a nie, jak w uproszczonym przykładzie, jednego połączenia), nie powoduje zatem dążenia do rozwiązania zachłannego.

3 Zaproponowane rozwiązania

3.1 TSP – Puris et al.

3.1.1 Zarys

W [2] autorzy prezentują algorytm PTS-ACO, będący rozwinięciem wcześniej opracowanego Two-Stage ACO, opisanego dokładniej w [3]. Idea polega na podziale poszukiwań najkrótszej ścieżki na dwie fazy. Każda z nich wykorzystuje rdzeń w postaci wybranego wariantu ACO. W pracy zbadana została użyteczność w tym celu modeli:

- AS – podstawowego, opisanego w sekcji 2.
- ACS – w którym podczas tworzenia rozwiązań mrówki "zjadają" część feromonu na wybieranych połączeniach, feromon dodawany jest tylko na najlepszej globalnie ścieżce, natomiast decyzja o następnym celu podejmowana jest w lekko zmodyfikowany sposób – z prawdopodobieństwem q wybierane jest połączenie o największym $\tau^\alpha \eta^\beta$, w przeciwnym wypadku postępuje się jak w AS. Pierwsze z tych podejść wspiera eksplorację, dwa pozostałe są silnie ekspansywne.
- MMAS – gdzie feromon dodaje tylko najlepsza mrówka w iteracji (ekspansja), a τ ma wartości maksymalną i minimalną (eksploracja).

Przez m oznaczmy liczebność populacji, t – liczbę iteracji wykonywania algorytmu, a przez n – liczbę miast. Wprowadzony zostaje nowy parametr, $r \in (0, 1)$, zgodnie z którym w pierwszej fazie użyjemy $m_1 = r \cdot m$ mrówek, a będzie ona trwać $t_1 = r \cdot t$ iteracji. W pojedynczym kroku tego etapu każda mrówka generuje $(r \cdot n)$-elementowe rozwiązanie częściowe. Sposób wybierania kolejnego kroku i zmiany poziomu feromonu zależne są od modyfikacji ACO, którą wybrano na rdzeń. Efektem fazy ma być wygenerowanie dobrego początkowego rozmieszczenia feromonu, które może pozwolić mrówkom na ominięcie mniej interesujących fragmentów przestrzeni i szybsze znalezienie dobrych rozwiązań.

W drugim etapie udział bierze $m - m_1$ pozostałych mrówek, a trwa on $t - t_1$ iteracji. W zasadzie przypomina tradycyjne ACO, a jedyną różnicą jest fakt, że na wejściu otrzymuje wyznaczone już wstępnie poziomy feromonu na poszczególnych połączeniach.

Celem wyboru rdzenia najbardziej odpowiedniego dla TSP, autorzy przeprowadzili na początku testy z pominięciem podejścia dwustopniowego. Okazało się, że najlepsze wyniki uzyskał wariant ACS, którego parametry zostały dobrane tak, żeby silna ekspansywność (zarówno w samej budowie metody, jak i w wysokim $\rho = 0.9$ oraz $q = 0.67$) została zrównoważona zmniejszeniem znaczenia feromonu w decyzjach mrówek ($\alpha = 1$ kontra $\beta = 5$), jak również wykorzystaniem przeszukiwania lokalnego na najlepszym rozwiązaniu w iteracji. Autorzy zdecydowali się też na $m = 10$ oraz początkowe dla pierwszej fazy $\tau_0 = 0.2$.

Twórcy wyszli z intuicyjnego założenia (motywowanego badaniami z [3]), że ten sam wariant najlepiej poradzi sobie też w podejściu dwustopniowym, dlatego został on zastosowany w obu fazach algorytmu. Interesującym i niezbadanym aspektem pozostają podejścia mieszane, czyli stosujące inny wariant w każdym z etapów. Najważniejszym parametrem PTS-ACO jest r, który określa, jak między fazami podzielone zostają zasoby i czas. Łatwo domyślić się, że niewielkie r są kontrproduktywne – $r = 0$ dałoby nam dokładnie podejście jednostopniowe, natomiast inne niskie wartości mogą okazać się zwykłą stratą iteracji (szczególnie gdy na przykład $r \cdot n \approx 1$). Zbyt wysokie r również może okazać się złym wyborem, ponieważ oznacza, że znaczna część zasobów zostanie wykorzystana w fazie inicjalizacji poziomów feromonu, natomiast sama eksploracja pełnych rozwiązań jest ograniczana. Z tego też powodu trudno jest dostroić parametr dla niewielkich, prostych instancji TSP. Spośród wartości $r \in \{0.2, 0.25, 0.3\}$ badania autorów wykazały znaczną przewagę dla największej z nich. Tak ograniczony zbiór uzasadniony jest wcześniejszymi badaniami w [3], które sugerowały największą efektywność algorytmu dla wartości z rozpatrywanego zakresu.

Mimo podobieństwa PTS-ACO do modelu zaproponowanego w pracy [3], chcąc stworzyć efektywną implementację, warto sprawdzić też inne – chociaż zbliżone do przedstawionych w badaniu – wartości dla r.

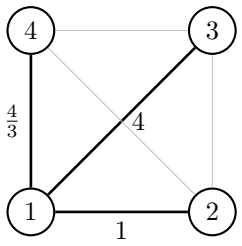
3.1.2 Przykład

Aby w czytelny sposób zilustrować ideę algorytmu, przyjmiemy rozmiar populacji $m = 2$, liczbę miast $n = 4$, czas trwania w iteracjach $t = 2$, podział etapów $r = 0.5$ oraz $q = 0.5$. Przykład ma na celu zaprezentowanie głównie dwustopniowego procesu, a niekoniecznie wszystkich szczegółów modyfikacji ACS, które wymagałyby użycia większej liczby mrówek oraz przyjrzenia się większej liczbie iteracji. Pozostałe parametry mają wartości takie jak w przykładzie z sekcji 2. Oznacza to, że pierwsza faza trwać będzie jedną iterację, w której jedna mrówka wybiera dwuelementową trasę. Przyjmijmy, że mrówka zaczyna w punkcie 1, wówczas faktyczne prawdopodobieństwa wyboru poszczególnych miast (z uwzględnieniem q, oznaczane $P_{i,j}$) wyznaczamy ze wzoru na prawdopodobieństwo całkowite:

$$p_{1,2} = \frac{1}{1 + 0.25 + 0.75} = 0.5 \qquad P_{1,2} = q \cdot 1 + (1 - q) \cdot p_{1,2} = 0.75$$

$$p_{1,3} = \frac{0.25}{1 + 0.25 + 0.75} = 0.125 \qquad P_{1,3} = q \cdot 0 + (1 - q) \cdot p_{1,3} = 0.0625$$

$$p_{1,4} = \frac{0.75}{1 + 0.25 + 0.75} = 0.375 \qquad P_{1,4} = q \cdot 0 + (1 - q) \cdot p_{1,4} = 0.1875$$



Przyjmijmy zatem, że mrówka wybrała przejście do 2. Ponieważ osiągnęła wymaganą długość rozwiązania, kończy pracę i uwalnia feromon:

$$\Delta\tau_{1,2} = 1$$

$$\tau_{1,2} = 0.5 \cdot 1 + 1 = 1.5$$

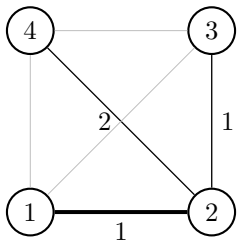
Zmienia się także poziom feromonu na każdym z pozostałych połączeń, mamy tam $\tau = 0.5 \cdot 1 = 0.5$.

Ustalone poziomy feromonu inicjują drugą fazę algorytmu. Przyjmijmy teraz, że mrówka wylądowała w punkcie 2. W tej części, ponieważ w przykładzie zostanie użyta tylko jedna mrówka na przestrzeni jednej iteracji, pominięte zostaną zmiany zachodzące w poziomie feromonu lokalnie (wspomniane wcześniej "zjadanie") oraz globalnie (po stworzeniu ścieżki).

$$p_{2,1} = \frac{1.5 \cdot 1}{1.5 + 0.5 + 0.25} = 2/3 \qquad P_{2,1} = q \ \cdot 1 + (1-q) \cdot p_{2,1} = 15/18$$

$$p_{2,3} = \frac{0.5 \cdot 1}{1.5 + 0.5 + 0.25} = 2/9 \qquad P_{2,3} = q \ \cdot 0 + (1-q) \cdot p_{2,3} = 2/18$$

$$p_{2,4} = \frac{0.5 \cdot 0.5}{1.5 + 0.5 + 0.25} = 1/9 \qquad P_{2,4} = q \ \cdot 0 + (1-q) \cdot p_{2,4} = 1/18$$



W przykładzie łatwo jest zauważyć ekspansywność nowego sposobu podejmowania decyzji – szansa na wybór spośród dwóch krawędzi o identycznej wadze tej o większym poziomie feromonu jest teraz ponad siedmiokrotnie większa, podczas gdy stara metoda dawałaby jej tylko trzykrotną przewagę. Mrówka ma bardzo dużą szansę na przejście do 1. W takim wypadku, ponieważ poziomy feromonu na dostępnych jeszcze krawędziach są identyczne, mrówka będzie mieć silną tendencję wyboru krawędzi krótszej (to jest przejścia do 4, jak na rysunku dla pierwszej fazy). Wówczas domyka kurs jedynym możliwym przejściem do 3, a następnie wraca do 2. Ponieważ skończył się czas działania algorytmu, znalezione rozwiązanie przyjmowane jest za ostateczne.

3.2 E-GTSP – Yang et al.

3.2.1 Zarys

W [4] zaprezentowany został algorytm znajdujący rozwiązania dla uogólnionego problemu komiwojażera. W podstawowej wersji algorytm w dużej mierze opiera się na schemacie opisanym w sekcji 2, a jedyną zasadniczą różnicą jest sposób wyboru kolejnego kroku. Przez V oznaczmy zbiór wszystkich miast, V_i to i-ta grupa (klaster) miast. Zbiór Ω_k, z którego k-ta mrówka może wybrać kolejny cel, wyznaczamy na podstawie wzoru

$$\Omega_k = \{j \in V : \ (\forall \ V_i \in T_k)(j \notin V_i)\}, \text{ gdzie}$$

$$T_k = \{V_i : \ \exists \ j \ \in V_i \text{ odwiedzone przez } k\text{−tą mrówkę}\}$$

Po utworzeniu Ω proces wyboru następuje według opisu z sekcji 2. Po odwiedzeniu wszystkich klastrów mrówka wraca na położenie startowe.

Autorzy rozpatrzyli też kilka modyfikacji mogących usprawnić działanie algorytmu. W pierwszej wzięto pod uwagę intuicję, że "bliskość" całej grupy do obecnego miasta powinna pozytywnie wpływać na wybór jej członków. Przez $g(i)$ oznaczmy grupę, w której znajduje się i-te miasto. Funkcja prawdopodobieństwa zmienia się na

$$p_{i,j} = \frac{\tau_{i,j}^{\alpha} \eta_{i,j}^{\beta} \cdot \pi_{i,g(j)}}{\sum_{h \in \Omega} \tau_{i,h}^{\alpha} \eta_{i,h}^{\beta} \pi_{i,g(h)}}, \text{ gdzie}$$

$$\pi_{i,V_j} = \frac{\sum_{m \in V_j} \tau_{i,m}^{\alpha} \eta_{i,m}^{\beta}}{\sum_{s \in \Omega} \tau_{i,s}^{\alpha} \eta_{i,s}^{\beta}}$$

Oznacza to, że grupa, w której najwięcej jest elementów "bliskich" aktualnemu położeniu, otrzymuje największą premię podczas podejmowania decyzji przez mrówkę.

Kolejną zmianą jest wprowadzenie mutacji, mechanizmu znanego z algorytmów genetycznych. Po wygenerowaniu rozwiązania losowo wybierane jest jedno z odwiedzonych miast. Usuwa się je, a następnie losuje inne, pochodzące z tego samego klastra, tak aby zachować poprawność trasy. Wylosowane miasto wstawiane jest w każdy możliwy slot (przy m grupach mamy oryginalnie ścieżkę o $m+1$ elementach, usuwając jedno miasto pozostaje nam m, a zatem

slotów pomiędzy dwoma miastami mamy $m-1$ – tyle też nowych rozwiązań generuje mutacja). Ostatecznym rozwiązaniem mrówki – a więc tym, na podstawie którego rozmieści feromon – jest ścieżka o najmniejszej długości (spośród oryginalnej i tych powstałych w wyniku mutacji). Prawdopodobieństwo zajścia opisanej mutacji określa parametr p_{mut}. Zbyt wysokie ustawienie może okazać się niekorzystne, szczególnie w przypadku dużej liczby grup, ponieważ ograniczymy w ten sposób liczbę iteracji właściwej pętli algorytmu. Ponowne tworzenie ścieżek jest istotą ACO i pozwala na dużo szerszą eksplorację przestrzeni niż proste zmiany w pojedynczym rozwiązaniu. Z drugiej jednak strony odpowiednio dobrana wartość parametru może umożliwić nam wyjście z lokalnego minimum, jak również przyspieszyć znalezienie najkrótszej ścieżki.

Ostatnią modyfikacją jest wprowadzenie lokalnego przeszukiwania 2-opt na znalezionym rozwiązaniu. Dla każdej pary punktów w rozwiązaniu dokonujemy inwersji podścieżki między nimi, o ile tylko przyniesie to zmniejszenie długości całej trasy. Ten segment algorytmu pozwala mu na szybsze zbieganie do optimum.

Wyniki badań zamieszczonych w pracy sugerują, że jednoczesne zastosowanie wszystkich tych metod pozwala uzyskać lepsze wyniki niż w przypadku wyboru tylko niektórych lub żadnej.

Metodą prób i błędów autorzy wybrali $p_{mut} = 0.05$. Warto zwrócić uwagę, że gwarantują oni skuteczność zastosowanych środków dla problemów o liczbie miast mniejszej od 200. Implementując algorytm, którego dane wejściowe zawierać będą mocniej ograniczoną liczbę grup, można by pokusić się o ewentualne zwiększenie tej wartości – tak, żeby uzyskać większą eksplorację przestrzeni, nie tracąc zbyt wiele czasu.

3.2.2 Przykład

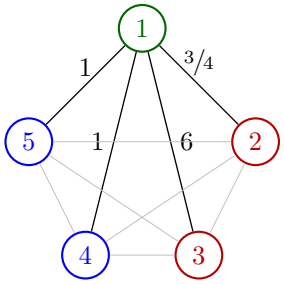
Przyjmiemy parametry takie jak w przykładzie z sekcji 2. Załóżmy, że mamy trzy grupy, nazwane od koloru, którym będą oznaczane – R, G oraz B. Przyjrzymy się działaniom pojedynczej mrówki, która w pierwszej iteracji rozpoczyna w punkcie 1.

$$T^1 = \{G\} \qquad \sum_{t \in T^1} \frac{\tau_{1,t}\pi_{1,g(t)}}{d_{1,t}} = (4/3 + 1/6) \cdot 3/7 + 2 \cdot 4/7 = 25/14$$

$$\Omega^1 = \{2,3,4,5\} \qquad p_{1,2} = \frac{4/3 \cdot 3/7}{25/14} = 8/25$$

$$\pi_{1,R} = \frac{4/3 + 1/6}{4/3 + 1/6 + 2} = 3/7 \qquad p_{1,3} = \frac{1/6 \cdot 3/7}{25/14} = 1/25$$

$$\pi_{1,B} = \frac{2}{4/3 + 1/6 + 2} = 4/7 \qquad p_{1,4} = p_{1,5} = \frac{1 \cdot 4/7}{25/14} = 8/25$$



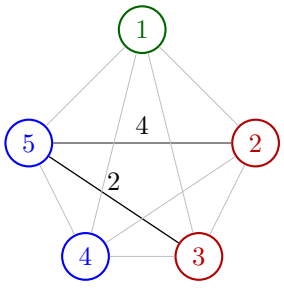
Jak widać, dodatkowy czynnik sprawia, że elementy grupy B możemy wybrać z taką samą szansą, jak 2, do którego prowadzi najkrótsza ścieżka. Przyjmijmy zatem taki scenariusz – mrówka przechodzi do 5, a następnie (ponieważ pozostała już tylko jedna grupa, a zatem $\pi_{5,R} = 1$), wybiera między czerwonymi miastami standardową metodą. Celem zaprezentowania mechanizmu mutacji, załóżmy życzeniowo, że podjęto decyzję o odwiedzeniu miasta bliższego (czyli 3, jak na rysunku poniżej). Mutacja może wówczas przebiec następująco:

$$len(1 \to 5 \to 3 \to 1) = 1 + 2 + 6 = 9$$

usuwamy 3, w zamian wstawiamy 2 w dostępne sloty:

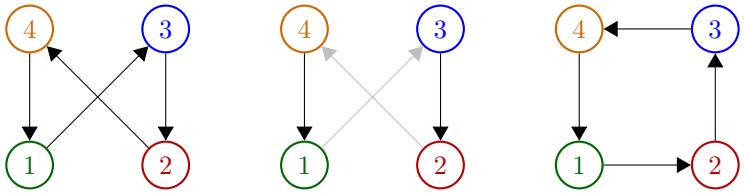
$$len(1 \to 2 \to 5 \to 1) = 0.75 + 4 + 1 = 5.75$$

$$len(1 \to 5 \to 2 \to 1) = 1 + 4 + 0.75 = 5.75$$



Dowolne ze zmutowanych rozwiązań zostaje wybrane jako ostateczne. Mutacja pozwoliła nam zatem na odkrycie ścieżki lepszej niż ta, którą początkowo wybrała mrówka.

Mechanizm inwersji jest trudny do zaprezentowania na rozpatrywanym przykładzie, dlatego zostanie zilustrowany odrębnie. Przyjmijmy, że miasta umieszczone są w rogach kwadratu, a długości ich połączeń odpowiadają ich odległościom euklidesowym, to znaczy krawędź będąca bokiem jest krótsza od tej będącej przekątną.



Trasa przedstawiona na pierwszym obrazku może być ulepszona poprzez inwersję cyklu między 2 i 3. Następuje zatem "przepięcie" – z miasta poprzedzającego cykl przechodzimy teraz do jego ostatniego elementu, natomiast z uprzednio pierwszego elementu cyklu wychodzimy do miasta, które go następuje. Ponadto odwrócona zostaje kolejność odwiedzania miast wewnątrz cyklu.

4 Podsumowanie

Opisane algorytmy, chociaż oparte na tej samej heurystyce i usiłujące rozwiązać ten sam problem, trudno jest ze sobą porównać, ponieważ ich twórcom przyświecały odmienne cele. Dwustopniowa metoda zaprezentowana w [2] dąży do ogólnego usprawnienia działania ACO, podczas gdy podejście opisane w [4] umożliwia rozszerzenie gamy problemów, które można zamodelować na podstawie TSP. Jednym z nich, wspomnianym w [1] jest rozmieszczenie skrzynek na listy, tak żeby skrócić trasę listonosza przy jednoczesnym zachowaniu odpowiedniej ich liczby na obszar.

Z drugiej jednak strony można zauważyć części wspólne. W obu artykułach zalecane jest dodatkowe lokalne przeszukiwanie wygenerowanych rozwiązań (czy to przez 2-opt, czy też mutacje). Te i inne sposoby pozwalają algorytmom na radzenie sobie z trudnościami napotkanymi podczas pracy z TSP. Zachłanność początkowych iteracji, w których poziomy feromonu są wyrównane, może doprowadzić do powstania bardzo nieoptymalnych rozwiązań. Ta tendencja jest balansowana przede wszystkim przez probablistyczny sposób podejmowania decyzji połączony z populacyjną naturą algorytmu. Mechanizmy przeszukiwania lokalnego mogą pomóc szczególnie w sytuacji, kiedy ostatni "przymusowy" krok rozwiązania – powrót na start – okaże się bardzo nieoptymalny. Stosowane są również dodatkowe wpływy na decyzje mrówek (odpowiednio parametr q oraz czynnik bliskości otoczenia π), pozwalające na efektywniejsze poruszanie się po mało zróżnicowanym pod względem wag grafie. Bardzo istotną kwestią w radzeniu sobie z problematycznymi instancjami w obu algorytmach jest odpowiednie dostrojenie zmian feromonu i jego początkowych poziomów, jak wspomniano w sekcji 2.

Na podstawie opisanych prac trudno jest jednak wyciągnąć wspólne wnioski na temat dokładnych wartości parametrów. Na przykład bardzo silnie ekspansywne rozwiązania użyte w PTS-ACS wymagają innego ich balansu niż łagodniejsze podejście opisane przy uogólnionym problemie. Znajomość wpływu parametrów i komponentów rozwiązania na jego cechy pozwala jednak istotnie zawęzić obszar poszukiwań.

Literatura

[1] Gilbert Laporte, Ardavan Asef-Vaziri, and Chelliah Sriskandarajah. Some applications of the generalized travelling salesman problem. *The Journal of the Operational Research Society*, 47(12):1461–1467, 1996.

[2] Amilkar Puris, Rafael Bello, and Francisco Herrera. Analysis of the efficacy of a two-stage methodology for ant colony optimization: Case of study with tsp and qap. *Expert Systems with Applications*, 37:5443–5453, 07 2010.

[3] Amilkar Puris, Rafael Bello, Yailen Martinez, and Ann Nowe. Two-stage ant colony optimization for solving the traveling salesman problem. In José Mira and José R. Álvarez, editors, *Nature Inspired Problem-Solving Methods in Knowledge Engineering*, pages 307–316, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[4] Jinhui Yang, Xiaohu Shi, Maurizio Marchese, and Yanchun Liang. An ant colony optimization method for generalized tsp problem. *Progress in Natural Science*, 18(11):1417 – 1422, 2008.