AKADEMIA GÓRNICZO-HUTNICZA

Wydział Informatyki, Elektroniki i Telekomunikacji

KATEDRA INFORMATYKI

Projekt z Badań Operacyjnych

Kierunek, rok studiów:	Semestr:	Rok akademicki
Informatyka, rok 3.	**VI**	**2015/2016**
Przedmiot:	**Unblock Me**	
Badania Operacyjne		
Prowadzący zajęcia:		
dr inż. Joanna Kwiecień		
Zespół autorski:	**Mikołaj Grzywacz** **Tomasz Kasprzyk** **Paweł Białas** **Daniel Ogiela**	Wtorek 14:40 grupa C

Spis treści

1. Wstęp (cel i założenia projektu)

Celem projektu jest bliższe zapoznanie się z jednym z algorytmów stadnych poprzez praktyczne wykorzystanie go w rozwiązaniu wybranego problemu. Na przykładzie wybranego algorytmu członkowie zespołu zrozumieją kluczowe idee, które kryją się za algorytmami stadnymi, poznają sposób projektowania i implementacji tego rodzaju algorytmu oraz zweryfikują przydatność algorytmów stadnych w praktycznych rozważaniach.

2. Opis zagadnienia:

2.1. Sformułowanie problemu

Problem polega na rozwiązaniu badanej gry w możliwie najmniej posunięciach.

2.2. Opis zagadnienia

“Unblock Me” (często spotykane pod nazwą “Unblock It”) to planszowa gra logiczna. Celem jest przestawienie wyróżnionego bloku (znajdującego w trzecim od góry wierszu dla tradycyjnej wersji (6x6); bloczek ten ma zwykle czewony kolor) na prawy koniec wiersza, gdzie znajduje się wyjście. Na drodze bloczka znajdują się inne, które musimy w odpowiedni sposób przestawiać, aby móc w końcu umieścić bloczek wyróżniony przed wyjściem.
Bloczki możemy podzielić ze względu na orientację w pionie i poziomie oraz ze względu na długość. Bloczki ustawione pionowo mogą poruszać się o dowolną długość (z dokładnością do wolnego miejsca i rozmiaru planszy) jedynie w pionie, analogicznie bloczki poziome. Każdy bloczek ma jeden z wymiarów równy dokładnie 1 (tak aby zajmował dokładnie jedną kolumnę/jeden wiersz). Rozmiary bloków pionowych mieszczą się w zakresie od 1x2 do (teoretycznie) wymiaru planszy, analogicznie bloki poziome.
Wynik rozgrywki nie zależy jedynie tego czy uda nam się umieścić czerwony bloczek

przy wyjściu. Punktowana jest także ilość ruchów, które pozwoliły nam osiągnąć cel. Im mniej, tym lepiej. Stąd pomysł na solver do gry, który pozwoli znaleźć odpowiedź na pytania: “Jaka jest najmniejsza ilość ruchów do osiągnięcia celu?” oraz “Jakie to są ruchy?”

2.3. Model matematyczny

a) Dziedzina problemu:

Bloczki występujące na kwadratowej planszy, które podlegają ściśle określonym regułom gry.

b) Istotne parametry:

- ilość przesunięć w rozwiązaniu – długość wektora rozwiązania
- wymiary planszy - w standardowej wersji problemu 6x6 (N=6)
- długości bloków: od 1x2 (2x) do 1x4 (4x1)
- wymiary wyróżnionego bloku: 1x2
- położenie początkowe wyróżnionego bloku: wiersz trzeci od góry (ułożony poziomo)

c) Określenie postaci rozwiązania

- Postać rozwiązania:
  solution = [{blockN, dir, step}, {block2, dir, step}, …, {blockN, dir, step}]
- blockN - identyfikator pojedynczego bloczka na planszy
- dir - kierunek przesunięcia bloczka
- step – ilość pól, o które trzeba przesunąć dany bloczek w danym kierunku

d) Funkcja celu

- Dziedzina - wektor rozwiązania
- Przeciwdziedzina - zbiór liczb naturalnych (określający liczbę kroków)

Funkcja celu sprawdza, czy dany wektor rozwiązania jest poprawny: w przypadku braku poprawności zwraca inf wpp. Zwraca długość wektora

e) Wprowadzenie w modelu uogólnień rozszerzających jego stosowalność
Rozmiar planszy może być modyfikowany - od 3x3 wzwyż (z uwzględnieniem mocy obliczeniowej i zasobów pamięci operacyjnej komputera / klastra komputerów. Możliwe jest również:

1. wybranie innego wiersza dla wyróżnionego bloku
2. Zmiana orientacji bloczka w pionie / poziomie (raczej mało użyteczne)
3. Zmiana długości bloczka
4. Zmiana położenia wyjścia (implikowane przez A. lub B.)

f) Zapisanie modelu w postaci matematycznej

- Macierz NxN - wartość pola informuje o obecności bloczka na danym polu

0	2	2	0	0
0	0	0	0	3
1	1	0	0	3
0	0	0	0	3
0	0	0	4	4

Przykładowa plansza 5x5

- Tablica rozmiaru N - przechowuje informacje o kierunku położenia bloczka oraz o jego długości

2.4.Zastosowania

Główne zastosowanie solvera to rozwiązanie określonego na wejściu układu bloczków na planszy w możliwie najmniejszej ilości ruchów.
W kwestii zagadnień realnych, solver możnaby stosować przy układaniu kontenerów / palet w magazynie. Kontenery, które powinny być wcześniej wykorzystane

znajdowałyby się w tych częściach magazynu, gdzie ilość przesunięć pozostałych magazynowanych obiektów jest jak najmniejsza.

3. Opis algorytmu: ogólny + szczegółowy opis adaptacji algorytmu

Algorytm docelowo stosowany przy rozwiązywaniu planszy to Ant Colony System.

3.1. Idea

Inspiracja dla algorytmu mrówkowego pochodzi ze świata mrówek, które potrafią znaleźć najkrótsza trasę między mrowiskiem a pożywieniem. Na początku procesu mrówki wyybierają trasę losowo w poszukiwaniu pożywienia, ale wracając do mrowiska pozostawiają na swojej trasie ślad feromonowy.
Feromony stopniowo parują.
Na najkrótszej trasie intesywność feromonu jest największa (z racji najkrótszego czasu parowania liczonego od rozpoczęcia powrotu przez mrówkę do dotarcia przez nią z powrotem do mrowiska).
W momencie znalezienia ścieżki prowadzącej do rozwiązania szybciej niż dotychczasowo utworzone trasy mrówki wybierają ją chętniej, wzmacniając ślad feromonowy. Jest to zjawisko pozytywnego sprzężenia zwrotnego- ścieżka bardziej atrakcyjna staje się jeszcze lepiej oznaczona feromonami - ścieżka gorsza (dłuższa) traci na znaczeniu (proces odnawiania ścieżki jest na tyle wolny, że feromony zdążają wyparować).

3.2. Schemat algorytmu podstawowego

- Wybierany jest rozmiar populacji mrówek
- Populacja mrówek jest tworzona na bazie jej rozmiaru
- Każda mrówka wybiera swoją ścieżkę niezależnie od innych mrówek
- Mrówka wybiera kolejne wierzchołki na bazie intensywności feromonu pozostawionego na krawędziach grafu, przy czym im więcej feromonu, tym większe prawdopodobieństwo wybrania ścieżki
- Mrówka wyposażona jest w pamięć, tak aby nie przechodzić dwukrotnie przez te same wierzchołki

- Po znalezieniu ścieżki mrówka pozostawia feromon na każdym odwiedzonym wierzchołku przy czym ilość pozostawionego feromonu zależy od efektywności znalezionego rozwiązania
- Algorytm jest iteracyjny - po każdej iteracji ta sama populacja mrówek szuka nowych rozwiązań

3.3. Adaptacja

a) Reprezentacja rozwiązania

Zgodnie z modelem matematycznym - lista zawierająca krotki {Identyfikator bloczka, ruch (w górę/ w prawo > 0 oraz w dół/ w lewo < 0)}, przykładowo [{#2, 1}, {#3, -1}, ..., {#1, 1}]

b) Inicjalizacja rozwiązań

Inicjalizacja algorytmem typu brute-force nie poprawia końcowego rezultatu, stąd wykonywany jest od początku algorytm mrówkowy.

c) Sposób wykonania ruchu

Ruch jest wykonywany według klasycznego rachunku prawdopodobieństwa. Pod uwagę brane są wszystkie możliwe przejścia. Im więcej feromonu na danym przejściu, tym większe prawdopodobieństwo wybrania tej trasy.

Prawdopodobieństwo przejścia dla danej mrówki ze stanu x do stanu y wynosi: $p_{xy} = \tau_{xy} / \Sigma_z \tau_{xz}$ gdzie τ_{xy} to wartość feromonu między stanami x i y. W mianowniku do sumy wliczamy wartości feromonów, które znajdują się na wszystkich możliwych przejściach od stanu x do innego.

Powstała również wersja, w której przy przejściu do kolejnego położenia brana jest pod uwagę odległość od końcowego stanu. Jednak oprócz eliminacji konieczności sortowania wyników mrówek po każdej iteracji nie wnosi znaczącej poprawy w uzyskiwanych rezultatach.

d) Odległość

Zdefiniowana jako liczba dozwolonych ruchów potrzebnych do przejścia w dany stan.

e) Sposób przeszukiwania sąsiedztwa

Przeszukiwane sąsiedztwo w odległości 1.

f) Naniesienie feromonu i jego parowanie

W pierwotnej wersji feromon był roznoszony przez każdą mrówkę w populacji. Wartość pozostawionego feromonu między kolejnymi stanami jest odwrotnie proporcjonalna do kwadratu długości rozwiązania. W ulepszonej wersji feromon jest naniesiony na przejścia tylko przez 10% najlepszych mrówek w danej populacji. Czynnik parowania jest określany w pliku konfiguracyjnym i opisuje jaka część feromonu zostanie na danym przejściu po częściowym odparowaniu feromonu. Wartość feromonu po przejściu całej populacji mrówek w danej iteracji wynosi: $\tau_{xy} = \rho^t * \tau_{xy} + \Sigma \Delta\tau_{xy}$, gdzie τ_{xy} - wartość feromonu między położeniami x i y, ρ - czynnik parowania, t - liczba iteracji od ostatniej obecności w niej mrówki, $\Delta\tau_{xy} = (pheremoneForAnt / solutionEfficiency)^2$

g) Pseudokod

```
def glowna_petla():
        while t < liczba_epok:
                rezultat = odpal_mrowki(liczba_mrowek)
                t += 1
        return rezultat

def odpal_mrowki(liczba_mrowek):
        dla każdej mrowki:
                while !gra_zakonczona:
                        wykonaj_ruch()
        nanieś_feromon_10%_najlepszych_mrówek()
        uwzglednij_parowanie _feromonu()


def wykonaj_ruch:
        znajdz_wszystkie_mozliwe_przejscia_w_odleglosci_1()
        wybierz_jedno_z_przejsc_wg_rach_pstwa()
```

h) Sposób doboru parametrów

Parametry dobrane eksperymentalnie.

4.Aplikacja – opis, dokumentacja użytkownika

4.1. Opis

Aplikacja stworzona w technologii Java. Graficzny interfejs użytkownika wykonany w technologii JavaFX. Aplikacja umożliwia wykonywanie obliczeń zarówno algorytmem typu brute-force, jak również algorytmem mrówkowym. Wybór algorytmu oraz ewentualne parametry do algorytmu mrówkowego określane w pliku konfiguracyjnym.

4.2. Dokumentacja użytkownika

Do uruchomienia aplikacji wymagane jest posiadanie środowiska wykonawczego Javy w wersji 1.8. Klasą uruchomieniową aplikacji jest klasa App.java. W celu uruchomienia programu należy wykorzystać plik *unblock-me-solver.jar*, który znajduje się w katalogu *executable* głównego katalogu projektu. Do folderu, w którym znajduje się archiwum jar, należy skopiować katalog *res/configs*. Ostatnim krokiem jest uruchomienie z linii poleceń komendy *java -jar unbloc-me-solver.jar arg*, gdzie *arg* to nazwa pliku konfiguracyjnego znajdującego się w katalagou *res/configs.*

Innym, niepolecanym sposobem jest wykorzystanie do uruchomienia aplikacji platform programistycznych (Eclipse, Intellij, itd.) bądź wykorzystanie polecenia *java* z odpowiedniego katalogu projektu.

Plik konfiguracyjny musi być w następującym formacie:

- Pierwsza linia to *BRUTE = FALSE* lub *BRUTE = TRUE.* W ten sposób określana jest odpowiednio niechęć bądź chęć do użycia algorytmu typu brute-force w obliczeniach
- Druga linia określa rozmiar planszy i jest w postaci: *SIZE = WYRAŻENIE,* gdzie *WYRAŻENIE* to liczba całkowita
- Kolejne linie pliku konfiguracyjnego zajmuje blok *GAMEBOARD_BEGIN ... GAMEBOARD_END,* między którymi znajduje się *SIZE* wierszów. W każdym wierszu znajduje się *SIZE* liczb całkowitych, które namierzają bloczki na planszy. 0 oznacza brak bloczka w tym miejscu, inna liczba oznacza, że w danym polu znajduje się część danego bloczka
- Ostatni blok to *ANT_SETTINGS_BEGIN … ANT_SETTINGS_END,* między którymi znajdują się w kolejnych liniach przypisania kolejnych parametrów do

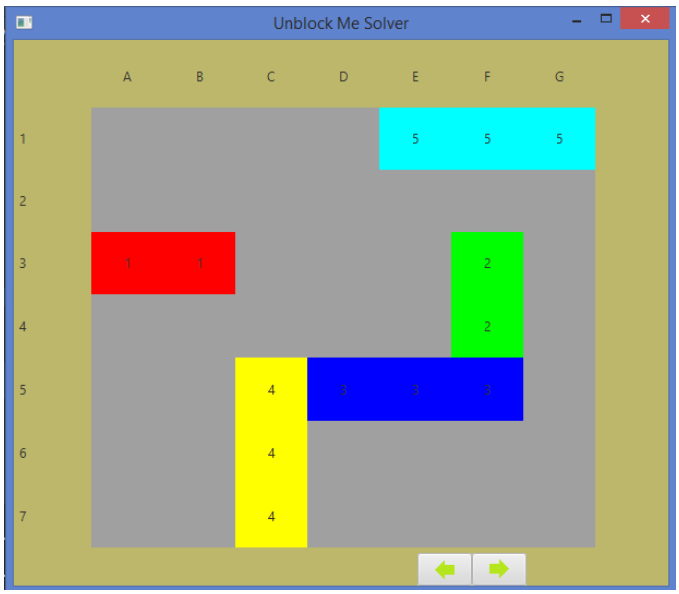
algorytmu mrówkowego. Są to kolejno: liczba mrówek, liczba epok, początkowa wartość feromonu, początkowa wartość epoki, czynnik parowania oraz wartość feromonu roznoszonego przez jedną mrówkę

```
BRUTE = FALSE
SIZE = 7
GAMEBOARD_BEGIN
    0 0 0 0 5 5 5
    0 0 0 0 0 0 0
    1 1 0 0 0 2 0
    0 0 0 0 0 2 0
    0 0 4 3 3 3 0
    0 0 4 0 0 0 0
    0 0 4 0 0 0 0
GAMEBOARD_END

ANT_SETTINGS_BEGIN
    ANT_QUANTITY = 500
    TRIALS = 1000
    DEFAULT_INITIAL_PHEROMONE = 0.1
    DEFAULT_INITIAL_TIMESTAMP = 0
    DECAY_FACTOR = 0.9
    PHEROMONE_FOR_ANT = 1
    TODO = ...
ANT_SETTINGS_END
```

Przykładowa zawartość poprawnie zmodyfikowanego pliku konfiguracyjnego

Po uruchomieniu programu i zakończenia pracy przez mrówki co następuje przy przejściu przez wszystkie epoki, na ekranie powinno wyświetlić się okno z graficzną reprezentacją planszy w stanie początkowym. Umieszczone w dolnej części okna przyciski, na których widnieją strzałki umożliwiają przejrzenie rozwiązania.



Przykładowe okno ukazane po wykonaniu obliczeń

5. Testy

5.1. Test na efektywność i zbieżność algorytmu dla zmian różnych parametrów

Część testów dla tego podpunktu wykonano w pierwotnej wersji algorytmu, w której wszystkie mrówki niezależnie od rezultatu kładły feromon w odpowiedniej ilości na swoich przejściach. Wyniki opisanych testów są zawarte w pliku Testy.xls.

Pozostałą część testów wykonano już według finalnej wersji algorytmu, w której jedynie 10 % najlepszych mrówek pozostawia feromony po każdej iteracji. Pewne parametry algorytmy w trakcie trwania testów były stałe. Ustalono liczbę epok na wartość 50. Testowano jedną planszę, której rozmiar wynosił 7x7. Grę można było na tej planszy zakończyć po trzech ruchach. Plik znajduje się w odpowiednim katalogu projektu (res/configs) a jego nazwa to *gameboard.config*. Przez większość prób początkowa wartość feromonu na każdym przejściu wynosiła 0.1, ilość feromonu przypadająca na jedną mrówkę wynosiła 1, czynnik parowania równał się 0.9 a liczba mrówek wynosiła 100. Parametry te zmieniały się jeżeli test był przeprowadzany ze względu na ten parametr.

Liczba prób w każdym teście wynosiła 10. Oprócz najgorszego i najlepszego wyniku odnotowywano wartość średnią.

Wpływ czynnika parowania				
Czynnik parowania	Najgorsza wartość	Najlepsza wartość	Wartość średnia	Najlepsze znalezione rozwiązanie
0.8	71	30	49.1	3
0.9	27	5	11.7	
0.99	7	3	5.1	
Wpływ liczby mrówek				
Liczba mrówek	Najgorsza wartość	Najlepsza wartość	Wartość średnia	Najlepsze znalezione rozwiązanie
100	16	4	9.2	3
200	12	8	9.5	
500	9	3	4.8	
Wpływ początkowej wartości feromonu				
	Najgorsza wartość	Najlepsza wartość	Wartość średnia	Najlepsze znalezione rozwiązanie
0.1	15	4	8.1	4
0.2	18	4	9.6	
0.5	15	4	8.9	
Wpływ ilości feromonu na mrówkę				
	Najgorsza wartość	Najlepsza wartość	Wartość średnia	Najlepsze znalezione rozwiązanie
0.8	15	5	9.7	
1.5	25	7	12.3	
2.0	26	7	16.2	

Wpływ współczynnika parowania okazał się niebagatelny i głównie on decydował o jakości i zbieżności rozwiązania. Wynik poprawia również możliwe wysoko dobrana liczba mrówek. Pozostałe dwa testowane parametry miały pewien zakres, w którym wyniki zbytnio nie odbiegały od siebie. Poza tym zakresem jakość rozwiązań nieco spadała.

5.2. Test na zbieżność algorytmu dla dużej ilości epok dla wybranych prób

Test przeprowadzono dla czterech zestawów parametrów przy czym zmieniano tylko dwa parametry: liczbę mrówek i czynnik parowania, które miały przy wcześniejszych testach największy wpływ na zbieżność i efektywność algorytmu. Zbadano wszystkie kombinacje parametrów ustawiając liczbę mrówek na 100 lub 500, natomiast czynnik parowania ustawiając na 0.9 lub 0.99. Docelowo liczba epok wynosiła 1000. Pozostałe parametry algorytmu były dobrane możliwie optymalnie tj. początkowa ilość feromonów na przejściach 0.1 oraz ilość feromonów na mrówkę 1.

Dla czynnika parowania równego 0.9 próby z liczbą mrówek równą zarówno 100, jak i 500 zakończyły się niepowodzeniem. Przy liczbie mrówek równej 100 wyniki w dość szybkim tempie się rozbiegły. W jednej z epok najlepsza mrówka miała katastrofalny wynik – 488. W 140 epoce program się zawiesił. Nie wszystkie mrówki były w stanie znaleźć rozwiązanie. Przy liczbie mrówek równej 500 wyniki były stosunkowo zbieżne, choć w pewnych iteracjach zdarzały się wyniki w okolicach 24. Ze względu na dużą liczbę mrówek i tym samym większe ryzyko „zgubienia się" którejś z nich, w epoce 98 program się zawiesił.

Dla czynnika parowania 0.99 i liczby mrówek 100 program zakończył swe działanie, choć pod koniec działania powoli wyniki się rozbiegały.

Wykres:

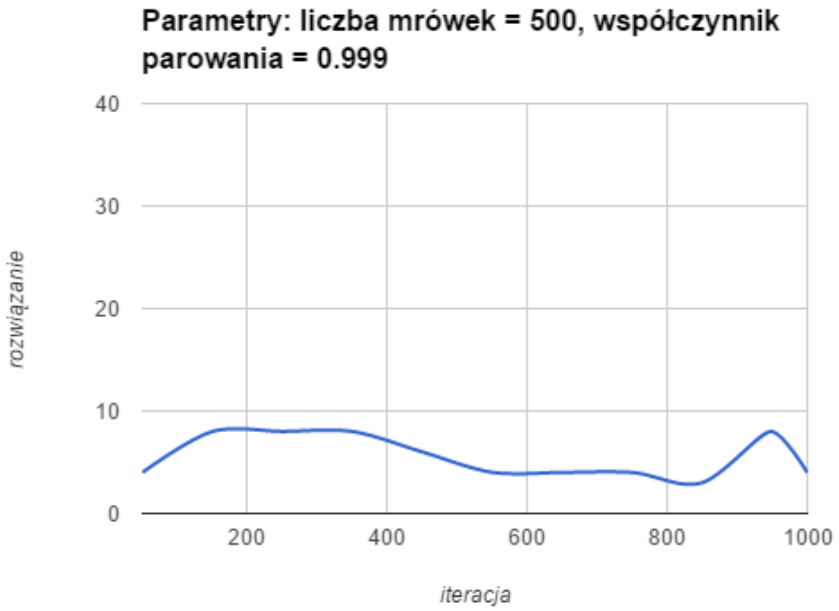
Dla czynnika parowania 0.99 i liczby mrówek 500 program zakończył swe działanie i zbieżność została zachowana na niezłym poziomie.

Wykres:

5.3. Test na zbieżność algorytmu dla optymalnych parametrów

W teście ustawiono następujące parametry: liczba mrówek – 500, początkowa wartość feromonu dla przejść – 0.1, współczynnik parowania – 0.999, ilość feromonu przypadająca na jedną mrówkę – 1.0. Parametry dobrano eksperymentalnie. Wykonano 1000 iteracji i algorytm pozytywnie przeszedł test zbieżności kończąc go z wynikiem 4.

Wykres:



6. Podsumowanie

Wybrana wersja algorytmu mrówkowego jest jedną z bardziej klasycznych odmian. Wybór kolejnego stanu zależy tylko i wyłącznie od ilości feromonów na przejściach, która wpływa na prawdopodobieństwo wyboru danego przejścia oraz od najbliższego sąsiedztwa. Uzupełnianie feromonów to również klasyczna odwrotność jakości rozwiązania, choć u nas podniesiono ją jeszcze do kwadratu. Zaimplementowany algorytm daje prawidłowe i dobre jakościowo wyniki.
Przy odpowiednim doborze parametrów algorytmu możliwe jest również uzyskanie zadowalającej zbieżności rozwiązań. Kluczowym parametrem naszego algorytmu okazał się współczynnik parowania feromonów, który do oczekiwanego działania musiał być wyjątkowo wysoki. Liczba mrówek przy dużej wartości również zwiększała efektywność działania algorytmu, jednak wpływ tego parametru był niewspółmiernie mały do współczynnika parowania. Udowodniły to testy na zbieżność. Dla średniej wartości współczynnika parowania nawet duże liczby mrówek nie zapobiegały rozbieżności wyników. Z kolei stosunkowo duża wartość współczynnika parowania zapewniała dość dobrą zbieżność

przy odpowiedniej liczbie mrówek. Pozostałe parametry potrzebowały mieć odpowiednią wartość by algorytm poprawnie działał. Nie pozostawiały w ten sposób dużego pola manewru.
Założenia projektu zostały spełnione. Zespół zapoznał się z ideą algorytmów stadnych na bazie konkretnego przykładu oraz nauczył się projektować i implementować algorytmy stadne. Jednocześnie zbadany problem zachęcił do dalszego zgłębiania wiedzy w tym kierunku.

7. Literatura

7.1. Wykład na temat algorytmów mrówkowych z Instytutu Informatyki Politechniki Poznańskiej [stan na 12.05.2016]
http://www.cs.put.poznan.pl/mradom/resources/labs/OptKomb/CI_wyklad_ewoluc_4.pdf

7.2. Clever Algorithms: Nature-Inspired Programming Recipes by Jason Brownlee PhD
http://www.cleveralgorithms.com/nature-inspired/swarm/ant_colony_system.html

7.3. Wykład z Badań Operacyjnych + laboratorium 2016
http://home.agh.edu.pl/~kwiecien/

7.4. https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms

7.5. Wykład na temat algorytmów mrówkowych z Uniwersytetu Wrocławskiego
https://www.ii.uni.wroc.pl/~prz/2011lato/ah/opracowania/alg_mrow.pdf