Katedra Wytrzymałości Materiałów
i Metod Komputerowych Mechaniki
www.kwmimkm.polsl.pl

Wydział Mechaniczny Technologiczny
Politechnika Śląska

# OBLICZENIA EWOLUCYJNE

## LABORATORIUM 6: Problem komiwojażera (TSP) – cz. 1

opracował: dr inż. Witold Beluch
witold.beluch@polsl.pl

Gliwice 2012

## Cel ćwiczenia

Wykonując ćwiczenia laboratoryjne zapoznasz się z zagadnieniem należącym do optymalizacyjnych problemów NP-trudnych w postaci tzw. problemu komiwojażera. Do jego rozwiązywania zastosujesz program optymalizacji ewolucyjnej *VG-TSP*. Zapoznasz się z działaniem programu i przetestujesz jego działanie dla różnych parametrów. Przeprowadzisz badania, w wyniku których określisz jak różne warianty kodowania osobnika i różne parametry algorytmu ewolucyjnego wpływają na skuteczność ewolucyjnych poszukiwań najkrótszej drogi.

## Trochę teorii

### Wprowadzenie

Problem komiwojażera (ang. *Traveling Salesman Problem*, *TSP*) został sformułowany jako zadanie matematyczne w latach 30-tych XX wieku, choć jego historia jest dużo starsza. Już w 1832 roku pewien podręcznik dla komiwojażerów wspominał to zagadnienie i zawierał przykładowe trasy uwzględniające Niemcy i Szwajcarię, choć bez opisu matematycznego problemu...

Problem komiwojażera jest określony następująco: jest dana (płaska) mapa, na której zaznaczone są miasta. Znane są odległości między miastami. Zadaniem komiwojażera jest wyjść z jednego miasta, odwiedzić wszystkie pozostałe – każde z nich wyłącznie jednokrotnie – oraz powrócić do miejsca startu, przy czym sumaryczna odległość, jaką pokonał, ma być najmniejsza z możliwych. Problem ten jest jednym z najbardziej znanych i najczęściej rozważanych problemów optymalizacyjnych.

Bardziej formalnie problem komiwojażera jest zadaniem poszukiwania w grafie pełnym[1] tzw. *cyklu Hamiltona*[2] o minimalnej sumie wag krawędzi („odległości"). Dowolny graf pełny posiada przynajmniej 1 cykl Hamiltona. Z faktu, że graf ma skończoną liczbę wierzchołków wynika, że w zbiorze cykli Hamiltona istnieje przynajmniej jeden taki, który posiada minimalna sumę wag krawędzi.

Problem komiwojażera jest NP-trudny, co oznacza, że nie są znane algorytmy o wielomianowej złożoności obliczeniowej rozwiązujące ten problem (przypuszczalnie takie nie istnieją). Ma on złożoność wykładniczą typu $O(n!)$ i dla $n$ miast liczba wszystkich kombinacji $k_n$ (przy założeniu symetrii problemu, czyli że droga w jedną i w drugą stronę ma taki sam koszt) wyraża się zależnością:

$$k_n = \frac{(n-1)!}{2} \tag{1}$$

Z powyższego wynika, że przykładowo zwiększenie liczby miast zaledwie z 60 do 61 skutkuje 60-krotnym wzrostem czasu obliczeń (czyli np. zamiast 1 minuty – 1 godzina). Mówimy w takim wypadku o *eksplozji kombinatorycznej* liczby możliwych dróg. Zagadnienia NP-trudne do poszukiwania zazwyczaj wymagają heurystyk, gdyż metody przeglądowe nie mogą być stosowane właśnie ze względu na eksplozję kombinatoryczną. Często stosowaną (meta)heurystyką[3] są algorytmy ewo-

[1] Graf pełny to taki, który łączy każdy wierzchołek z wszystkimi pozostałymi.

[2] Cykl (prosty) w grafie to ścieżka zamknięta – taka, której ostatni i pierwszy wierzchołek pokrywają się. Z kolei cykl Hamiltona to taki cykl, który przechodzi przez wszystkie wierzchołki, przy czym przez każdy z nich dokładnie jeden raz.

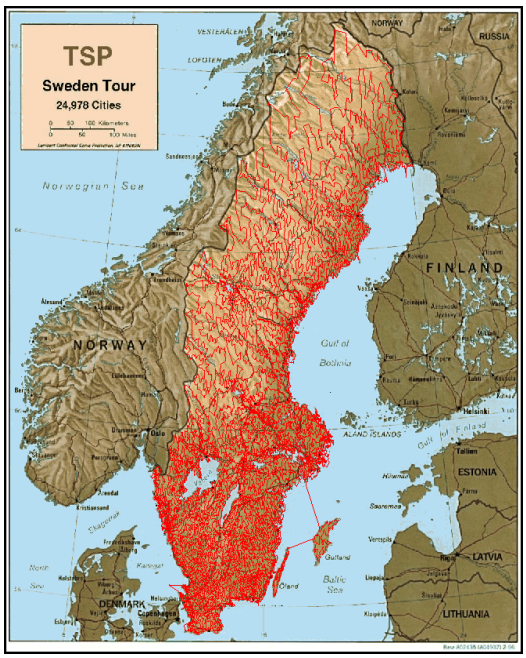
[3] Metaheurystyka wg. Wikipedii to „ogólny algorytm (heurystyka) do rozwiązywania problemów obliczeniowych. Algorytm metaheurystyczny można używać do rozwiązywania dowolnego problemu, który można opisać za pomocą pewnych definiowanych przez ten algorytm pojęć. Najczęściej wykorzystywany jest jednak do rozwiązywania problemów optymalizacyjnych. Określenie powstało z połączenia słowa "meta" ("nad", tutaj w znaczeniu "wyższego poziomu") oraz słowa "heurystyka" (gr. *heuriskein* - szukać), co wynika z faktu, że algorytmy tego typu nie rozwiązują bezpośrednio żadnego problemu, a jedynie podają sposób na utworzenie odpowiedniego algorytmu".

lucyjne, co wynika z możliwości stosowanie różnych reprezentacji problemu (sposobów kodowania zadania) i różnych operatorów ewolucyjnych, dopasowanych do specyfiki problemu. Największe w historii osiągnięcia w rozwiązywaniu TSP są przedstawione w Tab. 1.

Tab. 1 „Kamienie milowe" w historii TSP, za: http://www.tsp.gatech.edu

| Rok | Zespół badawczy | Wielkość zadania | Nazwa zadania w zasobach TSPLIB[1] |
|---|---|---|---|
| 1954 | G. Dantzig, R. Fulkerson, S. Johnson | 49 | dantzig42 |
| 1971 | M. Held, R.M. Karp | 64 | - |
| 1975 | P.M. Camerini, L. Fratta, F. Maffioli | 67 | - |
| 1977 | M. Grötschel | 120 | gr120 |
| 1980 | H. Crowder, M.W. Padberg | 318 | lin318 |
| 1987 | M. Padberg, G. Rinaldi | 532 | att532 |
| 1987 | M. Grötschel, O. Holland | 666 | gr666 |
| 1987 | M. Padberg, G. Rinaldi | 2 392 | pr2392 |
| 1994 | D. Applegate, R. Bixby, V. Chvátal, W. Cook | 7 397 | pla7397 |
| 1998 | D. Applegate, R. Bixby, V. Chvátal, W. Cook | 13 509 | usa13509 |
| 2001 | D. Applegate, R. Bixby, V. Chvátal, W. Cook | 15 112 | d15112 |
| 2004 | D. Applegate, R. Bixby, V. Chvátal, W. Cook, K. Heisgaun | 24 978 | sw24978 |

Na Rys. 1 przedstawiono mapę z zaznaczoną optymalną trasą o największej liczbie miast (Szwecja, 24 978 miast, 2004r), jaką dotychczas udało się znaleźć i udowodnić jej optymalność[2]. Znalezione rozwiązanie (http://www.tsp.gatech.edu/sweden/index.html) to trasa o długości 855 597 tzw. jednostek TSPLIB, co daje w przybliżeniu 72 500 kilometrów.




Rys. 1.Optymalna trasa dla 24 978 miast w Szwecji (2004),
Źródło: http://www.tsp.gatech.edu/sweden/tours/relief1.htm

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ - strona zawierająca wiele przykładowych zestawów miast dl problemu komiwojażera (nie tylko symetrycznego...).

[2] Jeśli chcesz się dowiedzieć, jak można stwierdzić (oczywiście nie sprawdzając wszystkich możliwości), czy znalezione rozwiązanie jest optymalne – zajrzyj na stronę http://www.tsp.gatech.edu/methods/opt/opt.htm.

Obecnym wyzwaniem jest trasa zawierająca 1 904 711 miast rozmieszczonych na całym świecie, co przedstawiono na Rys. 2. Dokładny opis problemu nazwanego *World TSP* (np. opis tego, w jaki sposób liczone są odległości między poszczególnymi miastami), można znaleźć na stronie http://www.tsp.gatech.edu/world/index.html). Jak dotychczas (maj 2012) najkrótsza znaleziona trasa ma długość 7 515 778 188 jednostek TSPLIB i została wyznaczona przez Kelda Helsgauna 25 października 2011 roku, czym pobił swój własny rekord z 4 kwietnia 2011 wynoszący 7 515 786 987 jednostek. Keld Helsgaun w swych poszukiwaniach korzystał z pewnego wariantu algorytmu heurystycznego o nazwie LKH[1].

Rys. 2. Położenie 1 904 711 miast w problemie TSP World

Przykładowe trasy TSP do przeprowadzania własnych poszukiwań najkrótszej drogi można znaleźć np. w TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/). W ramach niniejszego ćwiczenia skoncentrujemy się na ewolucyjnym podejściu do rozwiązania problemu komiwojażera.

## Kodowanie i operatory genetyczne

Problem komiwojażera ze względu na swą specyfikę wymaga specjalnego podejścia, jeśli do jego rozwiązania chcemy zastosować algorytm ewolucyjny. Pierwszym problemem, z jakim możemy się spotkać jest zagadnienie kodowania zadania – jak mianowicie zapisać „genetycznie" potencjalne rozwiązanie problemu, czyli osobnika?

Najczęściej stosowane są dwa sposoby kodowania (reprezentacje osobnika): reprezentacja ścieżkowa (*path*) i reprezentacja porządkowa (*ordinal*).

W **reprezentacji ścieżkowej**, zwanej też kodowaniem permutacyjnym, w osobniku zapisane są po prostu kolejne miasta, np. dla 7 miast dowolnie rozłożonych może to być: [**1 4 3 6 7 2 5**]. Zauważ, że osobnik w postaci [**6 7 2 5 1 4 3**] reprezentuje dokładnie to samo rozwiązanie (co niekoniecznie jest zaletą tego typu kodowania…).

Reprezentacja ścieżkowa wymaga specjalnych operatorów genetycznych – zastosowanie np. zwykłego krzyżowania jednopunktowego między dopuszczalnymi osobnikami: [**1 4 3 6** | **7 2 5**] oraz [**1 2 3 4** | **5 6 7**] powoduje, ze dostajemy dwa osobniki niedopuszczalne: [**1 4 3 6 5 6 7**] oraz [**1 2 3 4 7 2 5**] (powtarzające się miasta wyróżniono).

[1] LKH jest akronimem algorytmu wykorzystującego heurystykę Lina-Kernighana. Więcej informacji można znaleźć na stronie http://www.akira.ruc.dk/~keld/research/LKH/.

Istnieje wiele operatorów krzyżowania, które dla reprezentacji ścieżkowej dają dopuszczalne osobniki potomne w wyniku krzyżowania osobników dopuszczalnych, np:

- PMX – Partially Mapped Crossover (Goldberg, 1985);
- OX – Order Crossover (Davis, 1985);
- EX – Edge Crossover (Whitley, 1989);
- SXX – Subtour Exchanged Crossover (Yamamura, 1992)
- PX – Partition Crossover (Whitley, 2009)

I tak np. krzyżowanie typu PMX (krzyżowanie z częściowym odwzorowaniem) jest odmianą krzyżowania dwupunktowego. Weźmy 2 osobniki reprezentujące dwie trasy w problemie o 9 miastach:

[1 2 3 4 5 6 7 8 9] oraz [4 5 2 1 8 7 6 9 3]

Wybrano dwa punkty przecięcia: 3 i 7, czyli:

[1 2 3 | 4 5 6 7 | 8 9]
[4 5 2 | 1 8 7 6 | 9 3]

Zamieniamy części środkowe i na ich podstawie tworzymy tabelę odwzorowań:

[* * * | 1 8 7 6 | * *]
[* * * | 4 5 6 7 | * *]
1↔4, 8↔5, 7↔6

Następnie wstawiamy te miasta z osobników rodzicielskich, które nie powodują konfliktów:

[* 2 3 | 1 8 7 6 | * 9]
[* * 2 | 4 5 6 7 | 9 3]

W pozostałych miejscach wstawiamy, idąc od lewej strony, miasta zgodnie z tabelą odwzorowań. I tak np. w miejsce 1 w pierwszym osobniku rodzicielskim podstawiamy 4, w miejsce 8 podstawiamy 5 itd., otrzymując w efekcie:

[4 2 3 | 1 8 7 6 | 5 9]
[1 8 2 | 4 5 6 7 | 9 3]

W przypadku reprezentacji porządkowej stosuje się też dostosowane do jej specyfiki operatory mutacji, takie jak (pozostaniemy przy nazwach angielskich):

- *inversion mutation* – wybierająca losowo podciąg miast i odwracająca ich kolejność:

  [1 2 3 | 4 5 6 7 | 8 9] → [1 2 3 | 7 6 5 4 | 8 9]

- *insertion mutation* – przestawiająca losowo wybrane miasto na inną pozycję („rozsuwając" pozostałe geny):

  [1 2 3 4 5 6 7 8 9] → [1 7 2 3 4 5 6 8 9]

- *displacement mutation* – zamieniająca w wybranym losowo podciągu miast pierwsze miasto z ostatnim:

  [1 2 3 | 4 5 6 7 | 8 9] → [1 2 3 | 7 5 6 4 | 8 9]

- *transposition (exchange) mutation* – zamieniająca dwa losowo wybrane miasta:

  [1 2 3 4 5 6 7 8 9] → [1 6 3 4 5 2 7 8 9]

Zainteresowanych działaniem innych operatorów dla reprezentacji ścieżkowej odsyłam do literatury przedmiotu.

**Reprezentacja porządkowa** (*ordinal representation*) jest innym podejściem do kodowania trasy. Osobnik reprezentuje sobą kolejność, w jakiej z pewnej początkowej listy są wybierane miasta tworząc trasę. Kolejne liczby (geny) określają, które z pozostałych na liście miast należy wziąć jako następne na trasie, np.:

| | |
|---|---|
| lista miast: | ⟨**1 2 3 4 5 6 7 8 9**⟩ |
| i osobnik: | [**1 1 2 1 4 1 3 1 1**] |
| dają w efekcie trasę: | (**1 2 4 3 8 5 9 6 7**) |

Czyli: bierzemy z listy po kolei (zgodnie z informacjami zawartymi w osobniku): miasto nr 1, następnie pierwsze z pozostałych na liście, czyli miasto nr 2. Kolejnym miastem jest drugie z pozostałych po wykorzystanych już miastach 1 i 2, a tym jest miasto nr 4, itd.

Reprezentacja ta, mimo bardziej skomplikowanego podejścia i wymaganych pewnych operacji związanych z dekodowaniem osobnika, rekompensuje związane z tym problemy przy krzyżowaniu i mutacji. Cechą charakterystyczną tej reprezentacji jest to, że na *i*-tej pozycji jest liczba z przedziału od 1 do *n*-*i*+1 (gdzie *n* to liczba wszystkich miast). Z tego powodu wymiana materiału genetycznego między dwoma osobnikami za pomocą standardowego krzyżowania *x*-punktowego zawsze daje dopuszczalne osobniki potomne.

## Program *Visual Genetic – TSP*

Program *Visual Genetic – TSP* (VG-TSP) został napisany przez Pawła Tuszyńskiego w ramach pracy dyplomowej na Politechnice Krakowskiej w 2001 roku. Jak pisze autor:

*„Założeniem programu jest obserwacja działania algorytmów genetycznych dla problemów permutacyjnych szczególnie problemu komiwojażera. Program umożliwia dobór i testowanie różnych kombinacji operatorów genetycznych, jak i różnych parametrów algorytmu genetycznego. Mimo swej zewnętrznej prostoty, Visual Genetic posiada rozbudowane algorytmy z powodzeniem używane w ostatnich latach do sprostania wyzwaniom problemów permutacyjnych.*

*Jak sama nazwa wskazuje, dużą wagę przywiązano do wizualnego zobrazowania pracy algorytmu genetycznego, który mimo pewnej dawki losowości, zmierza stopniowo do rozwiązania optymalnego. Postępy algorytmu możemy śledzić w postaci tekstowej- jako ciągi kodowe populacji, graficznej – jako rozkodowaną trasę, po której porusza się komiwojażer jak i za pomocą różnego rodzaju wykresów. Do innych założeń uwzględnionych w programie należą:*

*- szybkość działania, mająca szczególne znaczenie przy dużych zadaniach;*

*- prostota interfejsu, która umożliwia korzystanie z programu również osobom, które nie posiadają dużej wiedzy o działaniu algorytmów genetycznych. Program jest łatwy w obsłudze. Wszystkie dane wprowadza się w standardowy dla takich programów sposób, a samo poruszanie się jest bardzo intuicyjne;*

*- przejrzystość kodu, która umożliwia dokładną analizę zastosowanych algorytmów jak i pozwala na szybką i łatwą rozbudowę programu w przyszłości o dodatkowe możliwości."*

## Zadanie

Rozwiąż zadanie komiwojażera dla zestawu miast w pliku o nazwie ***maze16.mst***. Początkowo przyjmij ustawienia domyślne programu *VG-TSP*, następnie zapoznaj się z różnymi możliwościami programu zgodnie z informacjami w dalszej części instrukcji.

Optymalne trasy dla wszystkich plików z miastami w katalogu *tours* są opisane w pliku *info.txt*.

## Zaczynamy

 Uruchom program *VG-TSP*. Włącz pomoc klikając odpowiednią ikonę. W pliku pomocy znajdziesz wszystkie potrzebne informacje na temat działania programu – w razie wątpliwości sięgnij właśnie tam. Poświęć trochę czasu na zapoznanie się z pomocą...

- Wciśnij w głównym oknie programu pierwszą ikonę z lewej strony lub wybierz *File*->*New Form*.
- Zapoznaj się z poszczególnymi zakładkami. W szczególności w zakładce *Data* otwórz odpowiedni plik z miastami. Nie zmieniaj na razie ustawień domyślnych. Wciśnij ikonę  . Jeśli algorytm zatrzyma się, naciskaj ikonę ponownie. Obserwuj okna programu. Jeśli chcesz, by algorytm był zatrzymywany co pokolenie, naciskaj ikonę  . Z kolei ikona  spowoduje zresetowanie zadania (ale nie ustawień zawartych w formularzu).

  Zauważ, że poszczególne okna potomne w programie (to, które są wyświetlane możesz ustawić w zakładce *Windows*) są „klikalne” prawym klawiszem myszy. Wypróbuj działanie poszczególnych opcji w tych oknach.

- Zmień warunek zatrzymania algorytmu. W tym celu otwórz formularz i wejdź do zakładki *Stop*. Wypróbuj działanie opcji. Zwiększ stosowne wartości według uznania - dla bardziej skomplikowanych zadań zazwyczaj domyślne wartości są za małe.

Próba zmiany jakichkolwiek ustawień w programie (otwarcie formularza) skutkuje zresetowaniem zadania…

- Przejdź teraz do zakładki *Operators*. Poeksperymentuj z operatorami genetycznymi, zaglądaj w razie wątpliwości do pomocy programu.
- Przejdź do zakładki *Task*. Zmień rodzaj reprezentacji. Zobacz, jakie operatory masz tym razem do wyboru.
- W zakładce Data znajduje się ramka *Initial tours*. Jeśli chcesz korzystać wyłącznie z algorytmu ewolucyjnego, pozostaw zaznaczone *Random*. Zobacz, jak działa algorytm z wybraną opcją *The closest neighbour algorithm*.
- Poeksperymentuj z programem...

W niektórych wersjach systemu Windows (np. Windows 7) nie działa funkcja tworzenia własnych plików z danymi (Zakładka *Data*->*Create File*). Jeśli jednak będziesz chciał stworzyć własną trasę, po prostu wygeneruj plik tekstowy o rozszerzeniu *.mst* w którym umieścisz współrzędne miast.

## Do wykonania

Przeprowadź badania dla pliku o nazwie **hand80.mst**.

- Opcję generowania trasy początkowej (*Initial tours*) koniecznie ustaw na *Random*.
- W zakładce *Task* ustaw sposób kodowania na ścieżkowy.
- W zakładce *Stop* ustaw: *Stop conditions*: *After* **500** *generations without improvement*.
- W zakładce *Operators* ustaw wstępnie:

  - *Elitist policy*: włączone;
  - *Selection: Fitness Function*: 1/*length*, *Scale fitness*: włączone, *ON*: zaznaczone;
  - *Crossover*: *Probability* ($p_k$): 0.5, *Type*: *PMX*, *Optimization*: *None*, *ON*: zaznaczone;
  - *Mutation*: *Probability* ($p_m$): 0.1, *Variable*: odznaczone, *Type*: *Inversion*, *ON*: zaznaczone

- Uruchom program. Zanotuj w tabeli 1 wyniki – długość znalezionej trasy oraz numer pokolenia, w którym algorytm się zatrzymał[1] – otrzymane po jednokrotnym naciśnięciu ikony  .
- Powtórz obliczenia dwukrotnie dla tych samych ustawień.
- Zmieniaj ustawienia zgodnie z danymi w tabeli 1 w protokole (pozostałe parametry pozostaw bez zmian) i wypełnij tą tabelę.
- Możesz poeksperymentować z różnymi innymi ustawieniami programu – zastosowane ustawienia i otrzymane wyniki zapisz w miejscu na notatki w protokole.

## Sprawozdanie

- Sprawozdanie ma być dostarczone wyłącznie w formie elektronicznej.
- Nazwa pliku wg wzorca: OE_lab6_Jan_Iksinski.doc/pdf.
- Strona pierwsza to strona tytułowa.
- W sprawozdaniu należy zamieścić:
  1. Cel ćwiczenia.
  2. Krótki opis problemu.
  3. Skan/fotografię protokołu.
  4. Opis przeprowadzonych badań i otrzymane wyniki.
  5. Wnioski z ćwiczenia.

## Literatura i źródła

[1] J. Arabas: Wykłady z algorytmów ewolucyjnych. WNT, Warszawa, 2003.

[2] Z. Michalewicz: Algorytmy genetyczne + struktury danych = programy ewolucyjne. WNT, Warszawa, 1996.

[3] http://www.tsp.gatech.edu/index.html - jedna z ciekawszych stron o TSP.

[4] http://www.mm.pl/~sielim/genetic/gen_komi.htm - *„Problem komiwojażera - przykład rozwiązania za pomocą AG”*.

[1] Oczywiście **nie** jest to numer pokolenia, w którym znaleziono najlepsze rozwiązanie...

# Protokół do laboratorium 6: Problem komiwojażera (TSP) – cz. 1

| Imię i nazwisko | Rok ak. | Gr. | Sem. | Komp. | Data | Podpis prowadzącego |
|---|---|---|---|---|---|---|
| _____ | 20__/__ | AB3 | I | _____ | _____ | |

Tabela 1. Parametry i wyniki dla zadania **hand80**.mst

| | Typ krzyżowania | $p_k$ | Typ mutacji | $p_m$ | Długość znalezionej trasy | Pokolenie zatrzymania |
|---|---|---|---|---|---|---|
| 1 | PMX | 0.5 | Inversion | 0.1 | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 1 | PMX | 0.8 | Inversion | 0.1 | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 1 | PMX | 0.2 | Inversion | 0.1 | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 1 | PMX | 0.5 | Transposition | 0.1 | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 1 | PMX | 0.8 | Transposition | 0.1 | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 1 | PMX | 0.2 | Transposition | 0.1 | | |
| 2 | | | | | | |
| 3 | | | | | | |

**Notatki** (tu i na drugiej stronie)**:**