Implementacja UCT algorytmu VF2 znajdowania izomorfizmu grafów

Maciej Bartoszuk

24.03.2016

Agenda

Czym jest izomorfizm grafów?

- Niech $G_1 = (V_1, E_1)$ oraz $G_2 = (V_2, E_2)$.
- Izomorfizmem dwóch grafów G_1 i G_2 nazywamy taką bijekcję pomiędzy V_1 i V_2

$$f : V_1 \to V_2$$

taką że dwa wierzchołki u i v w G_1 są połączone krawędzią wtedy i tylko wtedy gdy $f(u)$ i $f(v)$ są połączone krawędzią w G_2.

Algorytm VF2 – part 1

Algorytm opisany w [1, 2, 3, 4]

- Niech $M = \{(v_1, v_2) \in V_1 \times V_2 : v_1$ mapuje sie na $v_2\}$ będzie pewnym przyporządkowaniem wierzchołków z G_1 na G_2.
- W procesie mapowania jednego grafu na drugi możemy użyć przestrzeni stanów.
- Niech $M(s)$ to podzbiór M taki, że zawiera tylko niektóre pary wierzchołków. $M(s)$ indukuje nam także podzbiory grafów G_1 i G_2, oznaczmy je poprzez $G_1(s)$ i $G_2(s)$, uzyskane poprzez wybór tylko tych wierzchołków, które są w $M(s)$, a także krawędzie, które je łączą.
- Przejście pomiędzy stanami polega na dodaniu nowej pary zmapowanych wierzchołków.
- Aby ograniczyć liczbę stanów, które odwiedzamy, algorytm VF2 narzuca pewne ograniczenia na to, jaką parę wierzchołków możemy dołączyć do istniejącego rozwiązania. W szczególności $G_1(s)$ i $G_2(s)$ są względem siebie izomorficzne.

Algorytm VF2 – part 2

Początkowy stan s_0 taki, że $M(s_0) = \emptyset$

```
VF2(s, M(s))
{
 if(M(s) pokrywa wszystkie wierzcholki G2)
  zwroc rozwiazanie
 else
 {
  Oblicz zbior P(s) mozliwych par,
   ktore mozna dolaczyc do M(s)
  Dla kazdego p w P(s):
  if(jesli p spelnia zasady dolaczenia do M(s))
  {
   Oblicz stan s' poprzez dodanie p do M(s)
   VF2(s')
  }
 }
}
```

Algorytm VF2 – part 3

- Oznaczmy przez T_1^{out} i T_2^{out} wierzchołki, które jeszcze nie są w częściowym mapowaniu, ale do których wchodzą krawędzie wychodzące odpowiednio z $G_1(s)$ i $G_2(s)$.
- Oznaczmy przez T_1^{in} i T_2^{in} wierzchołki, które jeszcze nie są w częściowym mapowaniu, ale z których wychodzą krawędzie wchodzące odpowiednio do $G_1(s)$ i $G_2(s)$.
- $P(s)$ tworzony jest z par wierzchołków $v_1 \in T_1^{out}$ i $v_2 \in T_2^{out}$, jeśli te dwa zbiory nie są puste
- w przeciwnym wypadku $P(s)$ tworzony jest z par wierzchołków $v_1 \in T_1^{in}$ i $v_2 \in T_2^{in}$, jeśli te dwa zbiory nie są puste
- w przeciwnym wypadku $P(s)$ tworzony jest z par wierzchołków, które nie należą do $G_1(s)$ i $G_2(s)$ (sam początek algorytmu albo grafy niespójne)

Algorytm VF2 – part 4

- W grafie $G_2(s)$ wprowadzamy (dowolny) całkowity porządek na zbiorze wierzchołków. Algorytm ignoruje każdą parę (v_i, u_j), $v_i \in V_1$, $u_j \in V_2$ jeśli istnieje takie k, że w $P(s)$ $u_k \prec u_j$. Czyli tak naprawdę próbujemy parować dowolny wierzchołek z pierwszego zbioru z jednym, najmniejszym wierzchołkiem w drugim zbiorze, z którego wybieramy wierzchołki.

Algorytm VF2 – part 5

Zasady łączenia wierzchołków (u, v):

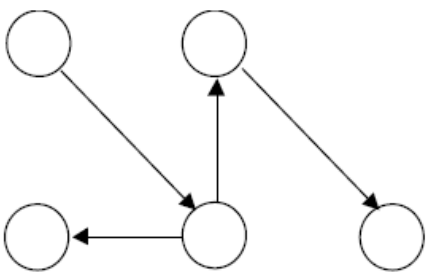
- Zmapowani sąsiedzi u pokrywają się ze zmapowanymi sąsiadami w drugim grafie oraz zmapowani sąsiedzi v pokrywają się ze zmapowanymi sąsiadami w pierwszym grafie
- Stopień wierzchołków się zgadza
- Liczba sąsiadów u, którzy są w $T_1^{in}(s)$ jest taka sama, jak liczba sąsiadów v, którzy są w $T_2^{in}(s)$. To samo tyczy się $T_1^{out}(s)$ i $T_2^{out}(s)$.
- Niech $\hat{V}_1(s) = V_1 - M_1(s) - T_1(s)$, gdzie $T_1(s) = T_1^{in}(s) \cup T_1^{out}(s)$. Analogicznie dla $\hat{V}_2(s)$. Liczba sąsiadów u, którzy są w $\hat{V}_1(s)$ jest taka sama, jak liczba sąsiadów v, którzy są w $\hat{V}_2(s)$.

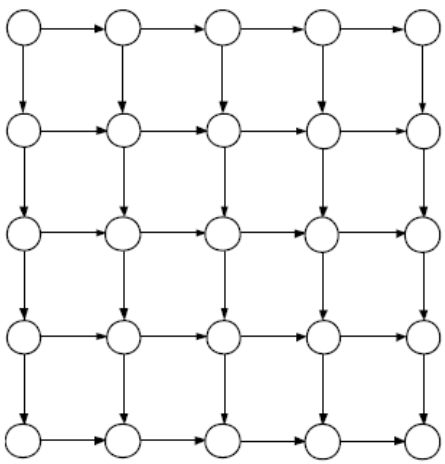
UCT

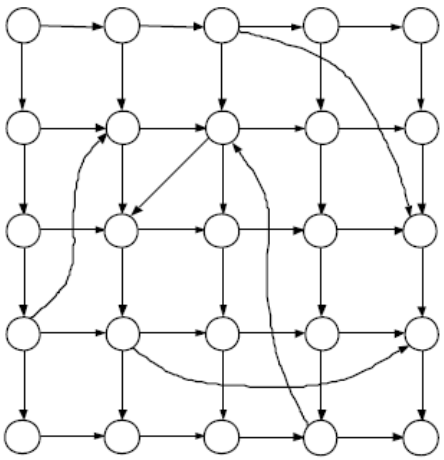
- Algorytm VF2 przegląda przestrzeń stanów w zgodnie z zasadą Depth-First Search (DFS). Naszym pomysłem jest użycie metody UCT, aby ukierunkować algorytm na przeglądanie przestrzeni stanów w pewnym określonym kierunku, który daje nadzieję na szybsze znalezienie izomorfizmu.
- Funkcją celu jest liczba znalezionych dopasowanych wierzchołków, ale rozważamy inne możliwości: wygładzanie funkcji celu, tak, aby mniejszą wagę przywiązywać do znikomej liczby wierzchołków, a większą do znacznej.
- Gdy znajdziemy ścieżkę, która ma większą wartość funkcji celu niż dotychczas znaleziona, to dołączamy ją do drzewa UCT.
- Jeśli istnieje ścieżka, która wiadomo, że nie ma żadnych rozgałęzień i już ją przebadaliśmy, to oznaczamy ją, aby drugi raz w nią nie wchodzić.
- Dopasowanie współczynnika c.

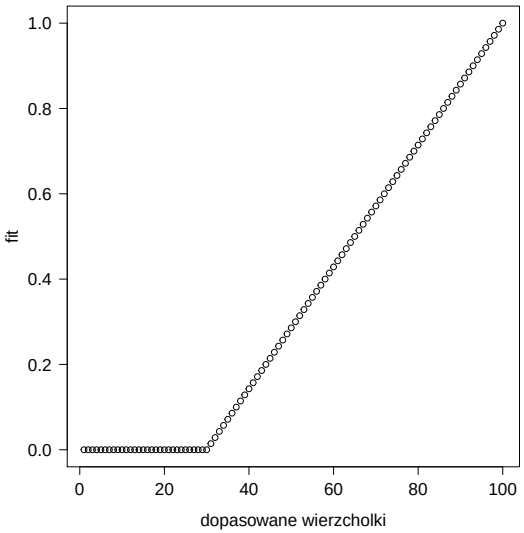
Baza opisana w [5].

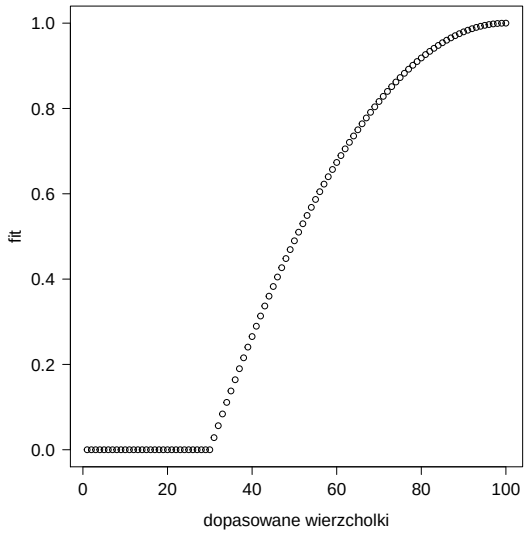
- *Randomly connected graphs*
- *Regular Meshes*
- *Irregular Meshes*

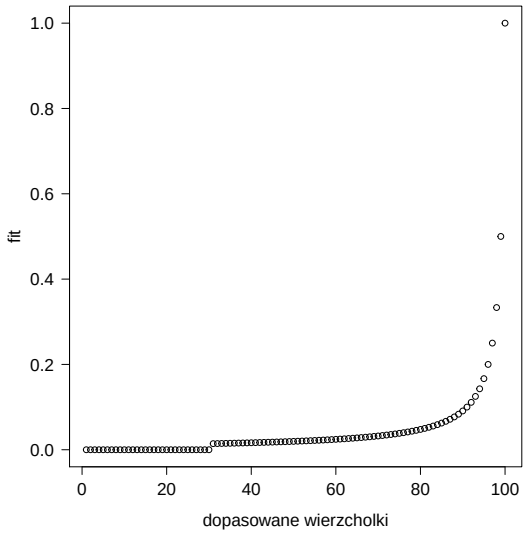


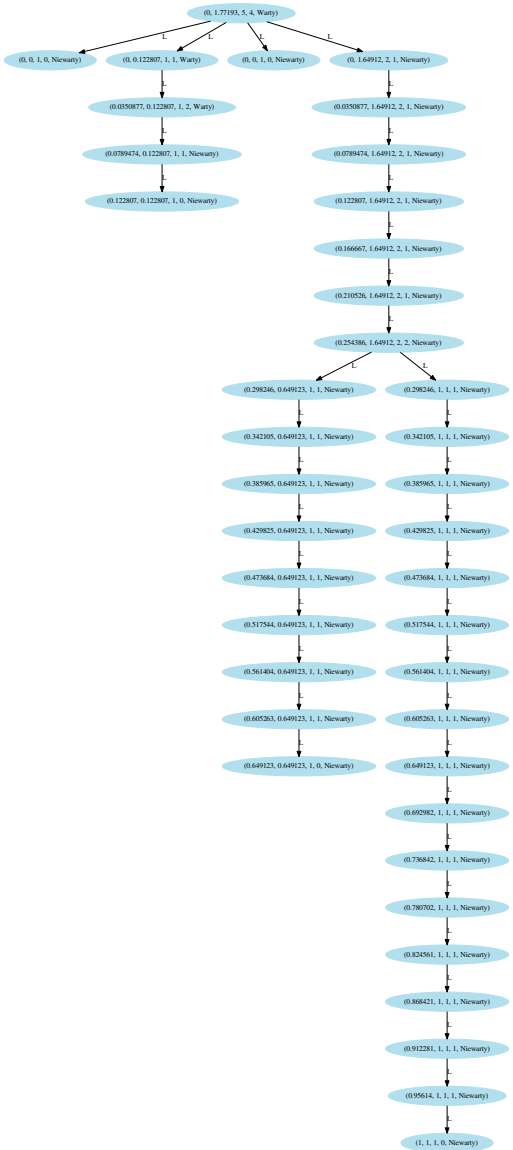





fit
dopasowane wierzcholki
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


fit
dopasowane wierzcholki
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


fit
dopasowane wierzcholki
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


(0, 1.77193, 5, 4, Warty)
(0, 0, 1, 0, Niewarty)
(0, 0.122807, 1, 1, Warty)
(0, 0, 1, 0, Niewarty)
(0, 1.64912, 2, 1, Niewarty)
(0.0350877, 0.122807, 1, 2, Warty)
(0.0789474, 0.122807, 1, 1, Niewarty)
(0.122807, 0.122807, 1, 0, Niewarty)
(0.0350877, 1.64912, 2, 1, Niewarty)
(0.0789474, 1.64912, 2, 1, Niewarty)
(0.122807, 1.64912, 2, 1, Niewarty)
(0.166667, 1.64912, 2, 1, Niewarty)
(0.210526, 1.64912, 2, 1, Niewarty)
(0.254386, 1.64912, 2, 2, Niewarty)
(0.298246, 0.649123, 1, 1, Niewarty)
(0.342105, 0.649123, 1, 1, Niewarty)
(0.385965, 0.649123, 1, 1, Niewarty)
(0.429825, 0.649123, 1, 1, Niewarty)
(0.473684, 0.649123, 1, 1, Niewarty)
(0.517544, 0.649123, 1, 1, Niewarty)
(0.561404, 0.649123, 1, 1, Niewarty)
(0.605263, 0.649123, 1, 1, Niewarty)
(0.649123, 0.649123, 1, 0, Niewarty)
(0.298246, 1, 1, 1, Niewarty)
(0.342105, 1, 1, 1, Niewarty)
(0.385965, 1, 1, 1, Niewarty)
(0.429825, 1, 1, 1, Niewarty)
(0.473684, 1, 1, 1, Niewarty)
(0.517544, 1, 1, 1, Niewarty)
(0.561404, 1, 1, 1, Niewarty)
(0.605263, 1, 1, 1, Niewarty)
(0.649123, 1, 1, 1, Niewarty)
(0.692982, 1, 1, 1, Niewarty)
(0.736842, 1, 1, 1, Niewarty)
(0.780702, 1, 1, 1, Niewarty)
(0.824561, 1, 1, 1, Niewarty)
(0.868421, 1, 1, 1, Niewarty)
(0.912281, 1, 1, 1, Niewarty)
(0.95614, 1, 1, 1, Niewarty)
(1, 1, 1, 0, Niewarty)

Bibliography I

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento.

An improved algorithm for matching large graphs.

In *In: 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen*, pages 149–159, 2001.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento.

Fast graph matching for detecting cad image components.

In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 1034–1037 vol.2, 2000.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento.

Performance evaluation of the vf graph matching algorithm.

In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 1172–1177, 1999.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento.

A (sub)graph isomorphism algorithm for matching large graphs.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(10):1367–1372, Oct 2004.

Bibliography II

P. Foggia, C. Sansone, and M. Vento.

A database of graphs for isomorphism and sub-graph isomorphism benchmarking.

In *CoRR*, pages 176–187, 2001.

Dziękuję za uwagę.