Porównanie algorytmów rysowania grafów

Piotr Piechal

14 marca 2011

Spis treści

Streszczenie

Dokument przedstawia opis i krótkie porównanie algorytmów służących do tworzenia i optymalizacji reprezentacji graficznej grafu. W porównaniu użyto prostego algorytmu planaryzacji oraz wybranych algorytmów sił skierowanych (zarówno dla małych jak i dużych grafów).

1 Prosty algorytm planaryzacji

Idea algorytmu zasadza się na sekwencyjnym dodawaniu kolejnych krawędzi do grafu planarnego. Jeśli dodanie krawędzi psuje własność planarności grafu, zaznaczamy ją jako nieplanarną. Następnie dodajemy kolejne krawędzie nieplanarne minimalizując liczbę przecięć. Algorytmy te mogą być dobrze sparametryzowane, przez co możemy określić na jakim aspekcie doskonałości grafu zależy nam najbardziej.

2 Algorytmy sił skierowanych

Algorytmy sił skierowanych (force-directed algorithms) charakteryzują się tym, że w całym procesie rozkładania grafu bazuje jedynie na danych zawartych w strukturze grafu. Grafy rysowane z wykorzystaniem tych algorytmów uważane są za estetyczne, odznaczają się wysoką harmonią i symetrią, a także cechuje je mała liczba przecięć krawędzi. Są zaprojektowane dla przetwarzania niewielkich grafów (do 50 wierzchołków) i mają dosyć dużą złożoność obliczeniową (rzędu $O(n^2 log(n))$). Poniżej omówione zostaną po krótce wybrane z algorytmów sił skierowanych. Zasada ich działania opiera się na minimalizacji funkcji energii wewnętrznej układu.

2.1 Tutte (1963)

Był to pierwszy algorytm sił skierowanych zaprojektowany by produkować prostoliniową reprezentację grafu bez przecięć krawędzi. Jego idea opiera się na zakotwiczeniu podgrafu planarnego i rozłożeniu pozostałych wierzchołków. Pozycje tych wierzchołków znajdujemy rozwiązując układ równań liniowych, w których położenie każdego wierzchołka jest wyrażone jako kombinacja liniowa położeń jego sąsiadów.

2.2 Eades (1984)

Jest uznawany za najbardziej tradycyjny algorytm z tej grupy. Jest zaprojektowany dla co najwyżej 30 wierzchołków i wykorzstje mechaniczny model dla osiągnięcia zamierzonych efektów. Jego idea opiera się na zamianie każdej krawędzi na sprężynę o idealnej długości 0, a wierzchołka na stalową obręcz, do której przyczepiane są sprężyny. Wierzchołki są ustawiane w jakiejś inicjalnej konfiguracji, a następnie siły naprężeń sprężyn dążą do minimalizacji energii układu. Najważniejszą cechą tego modelu jest zastąpienie sprężyn o charakterystyce liniowej (które okazały się zbyt silne dla bardzo oddalonych od siebie wierzchołków), sprężynami o charakterystyce logarytmicznej.

2.3 Kamada-Kawai (1989)

Modyfikcja algorytmu Eadesa z tą różnicą, że idealna długość sprężyn jest równa teoretycznej odległości wierzchołków w grafie (a nie 0 jak w pierwowzorze).

2.4 Fruchterman-Reingold (1990)

Modyfikacja algorytmu Eadesa z tą różnicą, że uwzględnia wierzchołki pierwotnie przywiązane. Wprowadzono także tak zwany współczynnik temperatury, który określa w jakim tempie mogą się przemieszczać wierzchołki

w kolejnych iteracjach, w zależności od tego, jak reprezentacja jest bliska optimum lokalnego.

2.5 Hadany-Harel (1999)

Jest to algorytm zaprojektowany dla rozkładania dużych grafów (powyżej 1000 wierzchołków). Jego idea polega na tym, że aby rozwiązać złożony problem, należy najpierw zgrubnie zbliżyć się do optymalnego rozwiązania, a następnie je poprawiać. To znaczy, że najpierw rozpatrujemy graf w pewnej skali, w której wiele wierzchołków jest grupowanych i rozpatrywanch jak jeden. Stsujemy doń algorytm Kamada-Kawai. W następnych krokach zwiększamy skalę i dzielimy wcześniej stworzone grupy na mniejsze aż do osiągnięcia skali pojedynczych wierzchołków. Twórcy algorytmu sugerują, by w tych krokach korzystać z algorytmów Eadesa, Fruchtermana-Reingolda oraz Kamada-Kawai.

2.6 Harel-Koren (2000)

Jest bardzo podobny do algorytmu Hadany-Harel, z tą różnicą, że korzysta z prostszej zasady podziału (na k centrów aproksymacji), a także z algorytmu Breadth-First Search. Do poprawiania lokalnych zaburzeń używa się algorytmu Kamada-Kawai. Algorytm ten został zaprojektowany dla bardzo dużych grafów (powyżej 15000 wierzchołków).

3 Porównanie

Algorytmy planaryzacji działają szybko, można je stosować do rozkładania różnych typów grafów, a wyniki ich stosowania są bardzo dobre nawet dla dużych grafów (w sensie ilości przecięć krawędzi). Niestety są bardzo skomplikowane w implementacji, a tworzone przez nie reprezentacje często różnią się od ludzkch wyobrażeń reprezentacji grafów. Algorytmy sił skierowanych są bardzo łatwe w implementacji, można w nich wprowadzać poprawki wynikające z heurystyki poza tym często tworzą symetryczne reprezentacje a działają świetnie dla małych grafów. Niestety i one nie są wolne od wad - mają długi czas działania.

4 Wnioski

Ponieważ diagramy UML nie są przykładami typowych grafów, konieczne będzie wprowadzenie pewnych modyfikacji do analizowanych algorytmów. W diagramach UML występują struktury, które zwyczajowo przedstawia się w określony sposób (np. dziedziczenie wielu klas po jednej macierzystej). Dlatego należy najpierw rozpoznać takie struktury, stworzyć ich reprezentację według zasad tworzenia diagramów UML, a w dalszym rozkładaniu

grafu traktować je jako pojedyncze wierzchołki (tak jak np. w algorytmie Hadany-Harel). Chciałbym zaimplementować kilka z przedstawionych powyżej algorytmów (Eades oraz Fruchterman-Reingold) a także algorytm prostej planaryzacji. Pozwoliłoby to na porównanie wyników działania tych algorytmów. Możliwe wyłoby wtedy również połączenie wyników działania obu typów algorytmów.

Literatura

[1] Roberto Tamassia (Brown University), *Handbook of Graph Drawing and Visualization*, CRC Press, Rozdział 12 *Force-directed drawing algorithms*

[2] Isabel F. Cruz (Worcester Polytechnic Institute), Roberto Tamassia (Brown University), *Graph Drawing Tutorial*