AUTOMATYKA • 2008 • Tom 12 • Zeszyt 3

Leszek Kotulski*, Adam Sędziwy*

Stochastyczne metody generacji IE-grafów

1. Wstęp

Transformacje grafowe często są stosowane jako mechanizm formalny w rozpoznawaniu obrazów [1, 2, 3, 4]. Dla algorytmicznych transformacji grafowych opartych na rozbudowie grafu w oparciu kontrolę jego wierzchołków lub krawędzi (*node lebel controlling*, *edge label controlling*) opracowano efektywne algorytmy parsingu i rozwiązywania problemu przynależności. Tanaka [5] w swoim przeglądzie stosowanych metod syntaktycznego rozpoznawania obrazów jako jedną z najbardziej rozwojowych wymienia gramatyki klasy ETL zdefiniowane przez Flasińskiego [6]. Dla kolejnej wersji gramatyki ETPL(k) powyższe problemy można rozwiązać z kwadratową złożonością obliczeniową $O(n^2)$, gdzie n jest liczbą wierzchołków w grafie. Przy dużych rozmiarach grafów jednak nawet tak dobra złożoność obliczeniowa realizowana w ramach algorytmu sekwencyjnego może prowadzić do nieefektywnych rozwiązań. W związku z powyższym podjęto więc próbę rozproszenia grafu i równoległej realizacji w ramach systemu wieloagentowego [7]. Dla zbadania własności proponowanej metody konieczne jest generacja różnych typów grafów (należących do klasy IE-grafów generowanej przez gramatykę ETPL(k)). W badaniach nad tworzeniem agentowych rozwiązań w zakresie rozpoznawania obrazów napotyka się na problem generacji dużych IE-grafów. W tym celu opracowano generator probabilistyczny IE-grafów, generujący grafy równoważne do tych, wykorzystywanych przez Flasińskiego w syntaktycznym rozpoznawaniu obrazów. Po omówieniu sposobu reprezentacji obrazów przy pomocy IE-grafów i wynikających stąd ograniczeń na ich generację (w rozdziale 2) opiszemy zasady działania generatora oraz przedstawimy charakterystyki generowanych grafów w zależności od metod generacji oraz parametrów generatora (odpowiednio w rozdziałach 3 i 4).

2. IE-grafy i ich zastosowanie

W niniejszym artykule chcemy przeanalizować podstawowe statystyczne własności struktur grafowych generowanych przez probabilistyczne generatory. Zakładamy jednak,

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie

że wygenerują one struktury porównywalne z tymi, które generowane są przez zastosowanie wcześniej ustalonych produkcji określonej gramatyki. Jak już wspomnieliśmy we wstępie, będziemy rozważali indeksowane, krawędziowo-jednoznaczne grafy, zwane dalej IE-grafami (*Indexed Edge-unambiguous*), z uwagi na fakt, że udało się dla nich [8]:

- zdefiniować schemat konstrukcji jednoznacznej i unikalnej grafowej reprezentacji rozważanej klasy obrazów,
- scharakteryzować taką podklasę gramatyk, dla której możliwy jest parsing deterministyczny,
- skonstruować algorytm analizy syntaktycznej charakteryzujący się niewielką złożonością obliczeniową.

Formalnie, indeksowanym, krawędziowo-jednoznacznym grafem, w skrócie IE-grafem, parametryzowanym przez Σ i Γ nazywamy piatkę $H = (V, D, S, G, \delta)$, gdzie:

V – jest skończonym, niepustym zbiorem wierzchołków grafu, którym zostały przypisane w sposób jednoznaczny indeksy na bazie obiektów analizowanego obrazu,

Σ – skończonym, niepustym zbiorem etykiet wierzchołkowych,

$\Gamma = \{\gamma_1, ..., \gamma_n: \gamma_1 \leq ... \leq \gamma_n\}$ – jest skończonym, niepustym zbiorem etykiet krawędziowych uporządkowanych przez relację prostego porządku,

D – jest zbiorem krawędzi postaci (v, μ, w), gdzie $w, v \in V$ i $\mu \in \Gamma$,

$\delta: V \rightarrow \Gamma$ – jest funkcją etykietowania krawędzi ■

Przykładowo, dla analizy zniekształconych obrazów [6] zbiór etykiet krawędziowych, które opisują relacje w przestrzeni dwuwymiarowej jest uporządkowany w następujący sposób:

$$N \leq NE \leq E \leq SE \leq S \leq SW \leq W \leq NW$$

IE grafy są transformowane za pomocą algorytmicznych transformacji grafowych bazujących na kontroli etykiet wierzchołkowych (*node labels control*) [4]. Dla zadanej produkcji w postaci dwóch grafów L i R (odpowiednio grafy lewej i prawej strony produkcji) oraz transformacji osadzenia E, modyfikacja grafu G przeprowadzona jest w sposób następujący:

- z grafu G usuwany jest graf izomorficzny do grafu L (ozn. m(L)),
- w miejsce usuniętego fragmentu wstawiany jest graf R,
- transformacja osadzenia E opisuje jak krawędzie łączące G\m(L) z m(L) zamienić na krawędzie łączące G\m(L) z R.

Dla zapewnienia wielomianowej złożoności parsingu i rozwiązywania problemu przynależności, na stosowane produkcje zostały przyjęte pewne ograniczenia. Już Rozenberg [4]

sugeruje, że bez zmniejszenia ogólności rozwiązania w stosowanych produkcjach lewa strona produkcji powinna być pojedynczym wierzchołkiem (wtedy izomorfizm sprowadza się wyłącznie do zbadania zgodności etykiet). W przypadku generacji grafu, złożoną produkcję z wielowierzchołkową lewą stroną można zastąpić przez sekwencję produkcji prostych (z jednowierzchołkowym grafem L). Flasiński dla uzyskania kwadratowej złożoności obliczeniowej w stosowanych grafowych (ETL/1 [6], ETPL(k) [2,9]) nakłada dodatkowe ograniczenie:

Generowany wierzchołek może być połączony krawędzią:

– z jego rodzicem (tj. wierzchołkiem, który inicjuje jego generację),
– z wierzchołkiem, który jest oddalony od niego o odległość 1 i jest połączony z jego ojcem.

3. Generacja IE-grafu

Algorytm działa w oparciu o wierzchołki i krawędzie będące obiektami o następującej strukturze.

– Wierzchołek (*Index*, *Pos*, *data*) – jest obiektem posiadającym unikalny indeks (*Index*), deskryptor położenia (*Pos*), będący zwykle parą współrzędnych kartezjańskich (x, y), oraz, opcjonalnie, dodatkowe informacje (*data*).
– Krawędź (*Index1*, *Index2*, *Direction*, *data*) – jest obiektem zawierającym indeksy wierzchołków końcowych (*Index1*, *Index2*), etykietę (*Direction*$\in$ {N, NE, E, SE, S, SW, W, NW}) opisującą kierunek wskazywany przez krawędź, oraz, opcjonalnie, dodatkowe informacje (*data*).

Stworzony algorytm działa w sposób iteracyjny: począwszy od pewnego wierzchołka początkowego (o polu *Index*=0) w kolejnych krokach dodawane są do budowanego grafu kolejne wierzchołki i/lub krawędzie. Algorytm działa w dwóch trybach *all-nodes-mode* (AM) oraz *FIFO-mode* (FM). Obie metody różni sposób wyboru węzła „startowego" w ramach kolejnej iteracji.

W podejściu AM wierzchołek „startowy" wybierany jest losowo, ze stałym prawdopodobieństwem, spośród wszystkich istniejących już węzłów grafu. W podejściu FM wykorzystywana jest kolejka FIFO, w której umieszczane są w kolejnych iteracjach wierzchołki nowo utworzone lub te, do których prowadzi dodana aktualnie krawędź. W przypadku gdy w danej iteracji nie doszło ani do wygenerowania nowego wierzchołka, ani nowej krawędzi, wówczas w kolejce umieszczony zostaje losowo wybrany istniejący już wierzchołek grafu.

Po wybraniu węzła „startowego" następuje etap polegający na wylosowaniu ilości krawędzi, m, jakie mają być dodane do węzła „startowego" danej iteracji. Losowanie to odbywa się zgodnie z rozkładem normalnym $N(\mu, \sigma)$. W następnym kroku wykonywane są m-krotnie subiteracje polegające na losowaniu (ze stałym prawdopodobieństwem) kierunku, na jakim zostanie dodana nowa krawędź (ze zbioru {N, NE, E, SE, S, SW, W, NW}) i nowy

wierzchołek (o ile jakiś już nie istnieje we wskazywanym położeniu). Zwrot krawędzi określony jest zasadą wynikającą z ograniczeń narzuconych przez gramatykę ETPL(k), przedstawioną niżej. Kiedy nie jest możliwe dodanie krawędzi na danym kierunku wykonywana jest kolejna subiteracja.

Przy generacji nowego wierzchołka w grafie, jego indeks ustawiany jest na zasadzie autoinkrementacji na wartość o 1 większą niż indeks poprzednio utworzonego wierzchołka.

Aby uzyskać zgodność ze strukturami grafowymi generowanymi przez gramatyki ETPL(k) [2, 6, 8, 9] grafy, które chcemy wygenerować, muszą spełniać następujące własności:

1) W kolejnym kroku iteracji nowa krawędź może być dodana w dwóch przypadkach:
   a) Jest to krawędź wchodząca do nowo utworzonego wierzchołka.
   b) Krawędź ta łączy dwa istniejące już wierzchołki posiadające wspólnego poprzednika.
2) Krawędź skierowana jest zawsze od wierzchołka o indeksie mniejszym do wierzchołka o indeksie większym.

Ostateczna topologia grafu zależy od dwóch głównych czynników: wyboru trybu działania algorytmu (AM/FM) oraz parametrów rozkładu normalnego μ, σ. Szczególnie istotny wkład wnosi wartość średnia μ decydująca o ilości nowych krawędzi dodawanych do wierzchołka w danej iteracji. Należy podkreślić, że stopień wierzchołka generowanego nie jest jednak prostą funkcją μ, gdyż na dodanie nowej krawędzi wpływa także istniejąca już topologia grafu.

Implementacja algorytmu zrealizowana została w języku JAVA przy użyciu biblioteki JUNG 2.0.

Deskryptory

Ocenę grafu otrzymanego z generatora dokonać można w oparciu o deskryptory dające jego charakterystykę. Charakterystyka powinna zapewniać zadowalające różnicowanie grafów, w zależności od wybranych parametrów generacji. Problematyka tworzenia i wyboru deskryptorów wykracza poza zakres niniejszej pracy, dlatego przedstawione zostaną tylko te, których użyto w badaniu własności IE-grafów otrzymywanych metodą generacji stochastycznej.

1) **Ilość krawędzi w grafie**.
2) **Średni stopień wierzchołka** – średnia liczona jest wspólnie dla krawędzi wchodzących i wychodzących, po wszystkich węzłach grafu.

$$\deg_{avg}(G) = \frac{1}{|V(G)|} \sum_{v \in V(G)} \deg(v).$$

3) **Mimośrodowość** – określa odległość średniego położenia wierzchołków od wierzchołka startowego (posiadającego indeks równy 0).

$$X_{avg} = \frac{1}{|V(G)|}\sum_{v \in V(G)} v_x, Y_{avg} = \frac{1}{|V(G)|}\sum_{v \in V(G)} v_y, v_{avg} = (X_{avg}, Y_{avg}),$$

$$Ecc = d_{Euclid}(v_{avg}, s),$$

gdzie Ecc oznacza mimośrodowość, d_{Euclid} jest odległością euklidesową położeń dwóch węzłów.

4) **Średnica** – maksymalna odległość (w sensie euklidesowym) między dwoma wierzchołkami grafu.

$$D = \max_{u,v \in V(G)} d_{Euclid}(u, v).$$

5) **Odległość maksymalna** – maksymalna odległość (w sensie euklidesowym) od węzła startowego.

$$d_{\max} = \max_{v \in V(G)} d_{Euclid}(s, v).$$

6) **Rozkład ilości węzłów** w zależności od odległości od wierzchołka początkowego – odległość w tym kontekście rozumiana jest jako ilość krawędzi od wierzchołka o indeksie 0, niezależnie od zwrotu krawędzi.

Przebieg testów

Testy polegały na 100-krotnym wygenerowaniu i obliczeniu deskryptorów IE-grafu o 3000 wierzchołkach, dla każdej z 4 konfiguracji parametrów wejściowych/metod (tab. 1).

Tabela 1
Użyte konfiguracje parametrów generacji IE-grafów

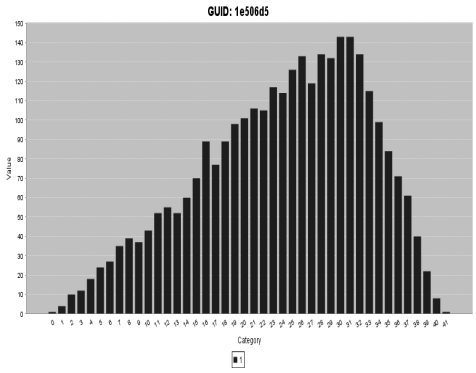
Seria 1	FM, μ = 1,5, σ = 0,2
Seria 2	FM, μ = 7, σ = 0,2
Seria 3	AM, μ = 1,5, σ = 0,2
Seria 4	AM, μ = 7, σ = 0,2

W tabeli 2 przedstawiono otrzymane wartości deskryptorów dla 100 przeprowadzonych generacji.

Tabela 2
Wyniki testów otrzymane dla różnych konfiguracji parametrów wejściowych. Deskryptory podano w postaci: wartość min./wartość średnia/wartość maks., przy czym wartości policzono dla 100 generacji

	Seria 1	Seria 2	Seria 3	Seria 4
$\|E\|$	3239/**3296**/3380	4248/**4346**/4431	4013/**4126**/4260	5015/**5136**/5325
$\deg_{avg}(G)$	2,16/**2,20**/2,25	2,83/**2,90**/2,95	2,68/**2,75**/2,84	3,34/**3,42**/3,55
Ecc	1,19/**6,42**/14,71	0,20/**2,87**/7,23	0,17/**2,24**/5,06	0,36/**2,42**/7,11
D	75,6/**88,3**/100,6	67,9/**73,2**/81,0	67,6/**72,7**/79,1	67,4/**72,2**/79,8
$d_{\max}$	39,0/**50,6**/63,5	36,1/**39,6**/47,2	34,9/**38,7**/44,7	35,1/**39,0**/45,3

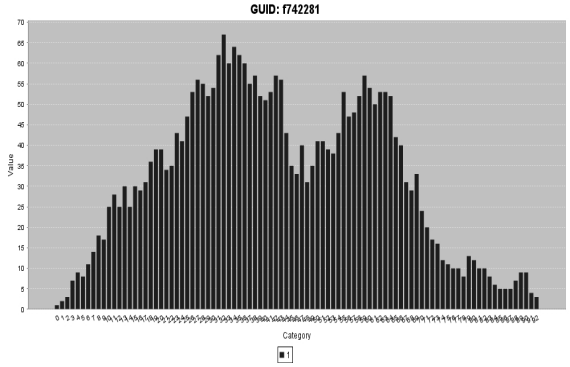
Ostatni deskryptor można interpretować jako rozkład mówiący o tym, w jakiej odległości od wierzchołka startowego *s* gromadzi się najwięcej wierzchołków. Inaczej mówiąc, w jakim „pierścieniu" o środku w *s*, jest najwięcej wierzchołków. Należy raz jeszcze podkreślić, że w tym przypadku mówimy nie o odległości euklidesowej, ale o ilości krawędzi dzielących dany wierzchołek od *s*.



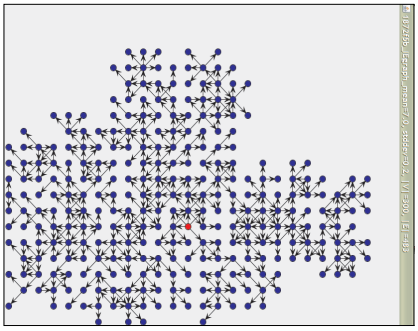
Rys. 1. Rozkład dla serii 2, 3, 4

Rysunek 1 pokazuje otrzymany rozkład, charakterystyczny dla serii 2–4. Testy pokazują, że rozkłady dla serii 2–4 charakteryzują się podobnymi kształtami, a w każdym z nich występuje pojedyncze maksimum. Odmiennie zachowują się grafy generowane w serii 1, gdzie występują przypadki dwóch, wyraźnie rozdzielonych maksimów lokalnych, odpowiadających zagęszczeniom w dwóch różnych „pierścieniach" (rys. 2).

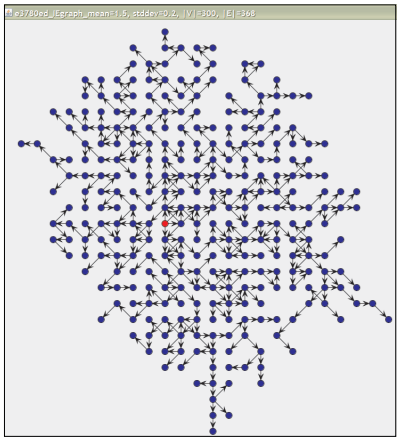
Grafy o najgęstszej strukturze uzyskuje się poprzez zastosowanie podejścia AM i wysokiej średniej ilości dodawanych krawędzi μ (rys. 3). W takim przypadku, w grafie wzrasta ilość 4-klik oraz 4-klik z usuniętą jedną bądź dwoma krawędziami. Rysunki 4 i 5 przedstawiają reprezentantów grup grafów 300 wierzchołkowych, generowanych przy konfiguracji jak w serii 3, oraz dla konfiguracji serii 1.



Rys. 2. Rozkład dla serii 1

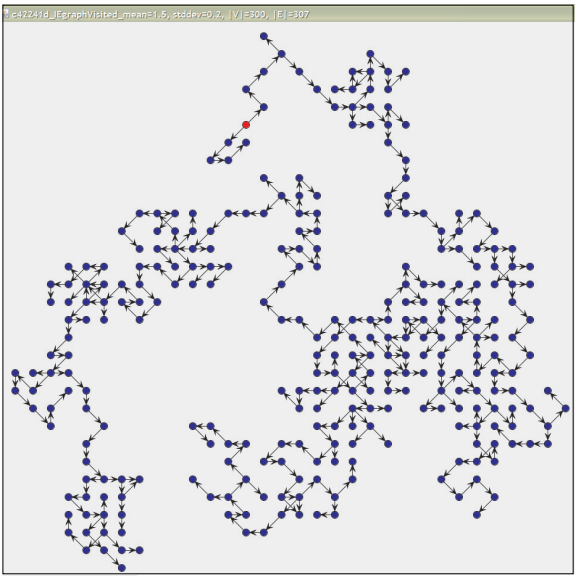


Rys. 3. Graf – seria 4. Na czerwono zaznaczono wierzchołek startowy



Rys. 4. Graf – seria 3. Na czerwono zaznaczono wierzchołek startowy

Na rysunku 5 widoczne jest, że graf posiada rzadszą strukturę, co wynika zarówno z niższej wartości parametru μ, jak i strategii dodawania nowych wierzchołków, polegającej na przyłączaniu ich do ostatnio dodanych węzłów grafu (FM).



Rys. 5. Graf – seria 1. Na czerwono zaznaczono wierzchołek startowy

4. Wnioski

Przedstawiony w poprzednim rozdziale stochastyczny generator IE-grafów stanowi efektywne narzędzie generacji grafów o żądanych właściwościach w zakresie struktury. Metodą opisu badanych własności jest analiza deskryptorów, pozwalająca na ilościowe porównanie otrzymanych grafów. W przeprowadzonych testach otrzymano topologie zależne zarówno od metod jak i parametrów generacji. Dwoma krańcowymi klasami wygenerowanych grafów są grafy rzadkie o małej liczbie krawędzi oraz dużej średnicy i mimośrodowości oraz grafy gęste (w klasie IE-grafów) charakteryzujące się zwartą i względnie symetryczną strukturą.

Generowane grafy o różnych charakterystykach pozwolą na ocenę algorytmów podziału grafu głównego na podgrafy, których transformacje będą odbywały się w sposób równoległy. Może mieć to istotne znaczenie nie tylko w przetwarzaniu obrazów [7], ale również w wielu innych zastosowaniach ponieważ IE-grafy z powodzeniem były zastosowane w takich zagadnieniach jak rozproszona kontrola alokacji [10, 11], formalnym opisie refactoring [12] czy systemach wieloagentowych [13].

Literatura

[1] Flasiński M., *Characteristic of edNLC-graph Grammars for Syntactic Pattern Recognition*. Computer Vision, Graphics and Image Processing, vol. 42, 1989, 1–21.
[2] Flasiński M., *On the Parsing of Deterministic Graph Languages for Syntactic Pattern Recognition*. Pattern Recognition, vol. 26, 1993, 16–93.
[3] Ogiela L., Tadeusiewicz R., Ogiela M.R., *Cognitive Computing in Intelligent Medical Pattern Recognition Systems*. [w:] De-Shuang Huang, Kang Li, Irwin G.W. (Eds), Intelligent Control and Automation, Lecture Notes in Control and Information Sciences, vol. 344, Berlin – Heidelberg – New York, Springer-Verlag 2006, 851–856.
[4] Rozenberg G., *Handbook of Graph Grammars and Computing by Graph Trans-formation: Volume I*. Foundations. Ed. World Scientific Publishing Co., NJ, 1997.
[5] Tanaka E., *Theoretical Aspects of Syntactic Pattern recognition*. Pattern Recognition, vol. 28, no. 7, 1995, 1053–1061.
[6] Flasiński M., *Distorted Pattern Analysis with the Help of Node Label Controlled Graph Languages*. Pattern Recognition, vol. 23, no. 7, 1990, 765–774.
[7] Kotulski L., *Distributed Graphs Transformed by Multiagent System*. Proceedings of the 9th International Conference on Artificial Intelligence and Soft Computing ICAISC 2008, Zakopane, Poland, Lecture Notes in Artificial Intelligence.
[8] Flasiński M., *Strukturalna analiza obrazów za pomocą gramatyk grafowych klasy ETPL(k)*. Rozprawy Habilitacyjne nr 233, Uniwersytet Jagielloński, 1992.
[9] Flasiński M., *Power Properties of NCL Graph Grammars with a Polynomial Membership Problem*. Theoretical Computer Science, vol. 201, 1998, 189–231.
[10] Flasiński M., Kotulski L., *On the Use of Graph Grammars for the Control of a Distributed Software Allocation*. The Computer Journal, vol. 35, 1992, A167–A175.
[11] Kotulski L., *Parallel Allocation of the Distributed Software using Node Label Controlled Graph Grammars*. Automatyka (półrocznik AGH), t. 12. z. 2, 2008.
[12] Kotulski L., Nowak A., *Formalizing Software Refactoring in the Distributed Environment by aedNLC Graph Grammar*. SET06 IFIP Conference, Warszawa 2006, in Software Engineering Techniques: Design for Quality – Springer Series in Computer Science, ISBN: 10:0-387-39387-0, 349–360.
[13] Kotulski L.. *Supporting Software Agents by the Graph Transformation Systems*. V.L. Alexandrow *et al.* (Eds), ICCS 2006, LNCS 3993, Reading (UK), 887–890.