Samostabilizujący się algorytm kolorowania grafów dwudzielnych i kaktusów

A self-stabilizing algorithm for coloring bipartite graphs and cacti

Adrian Kosowski i Łukasz Kuszner

{kosowski, kuszner}@eti.pg.gda.pl

Politechnika Gdańska, Katedra Algorytmów i Modelowania Systemów

Streszczenie

W artykule podano algorytm rozproszonego, samostabilizującego się kolorowania grafów. Rozważamy spójny system niezależnych, asynchronicznych węzłów, z których każdy posiada tylko i wyłącznie lokalną wiedzę o systemie. Bez względu na stan początkowy system powinien osiągnąć pożądany stan globalny wykonując w każdym z węzłów algorytm dany w postaci zbioru reguł. Zgodnie z naszą wiedzą przedstawiony algorytm jest pierwszym samostabilizującym algorytmem dokładnego kolorowania grafów dwudzielnych działającym w wielomianowej liczbie ruchów.

In the paper a distributed self-stabilizing algorithm for graph coloring is given. We consider a connected system of autonomous asynchronous nodes, each of which has only local information about the system. Regardless of the initial state, the system must achieve a desirable global state by executing a set of rules assigned to each node. Our method based on spanning trees is applied to give the first (to our knowledge) self-stabilizing algorithms working in a polynomial number of moves, which color bipartite graphs with exactly two colors.

1 Wstęp

Algorytmy samostabilizujące się rozważamy w systemach rozproszonych. Różnią się one od innych algorytmów rozproszonych w swej filozofii zapewniania poprawności. Podczas gdy tradycyjnie stosujmy sumy kontrolne, czasem skomplikowane schematy potwierdzania i retransmisji, to tutaj konstruujemy

algorytm tak, by z jakiegokolwiek nieprawidłowego stanu po awarii system powrócił w skończonym czasie do stanu poprawnego. Nie zakładamy więc nic o stanach początkowych, czy inicjalnych wartościach zmiennych [2].

System rozproszony modelujemy grafem $G = (V, E)$, gdzie wierzchołki ze zbioru V identyfikujemy z jednostkami obliczeniowymi, a krawędzie ze zbioru E odpowiadają połączeniom komunikacyjnym pomiędzy nimi. Zmienne lokalne każdej jednostki determinują jej *stan*. Sumę mnogościową stanów lokalnych poszczególnych jednostek nazywamy *stanem globalnym* systemu. Spośród wszystkich możliwych stanów globalnych wyróżniamy podzbiór *stanów legalnych*. Wykonanie algorytmu polega na krokowych zmianach stanów poszczególnych wierzchołków. Zmianę stanu jednego wierzchołka, czyli inaczej mówiąc zmianę wartości jego zmiennych lokalnych, będziemy nazywać *ruchem*. Pożądaną cechą algorytmów samostabilizujących jest doprowadzenie systemu do stanu legalnego przy jak najmniejszej liczbie ruchów.

Algorytm dla każdego wierzchołka u jest dany za pomocą reguł postaci: **if** $p(u)$ **then** M, gdzie akcja M opisuje ruch, a p jest warunkiem dotyczącym stanu wierzchołka u oraz stanów wierzchołków z *otwartego sąsiedztwa* wierzchołka u, $N(u) = \{v \mid \{v, u\} \in E\}$. Jeśli warunek p dla wierzchołka u jest spełniony, to mówimy, że wierzchołek jest *aktywny*. Przyjmujemy, że istnieje *demon*, który spośród wierzchołków aktywnych wybiera jeden, który jako kolejny wykona ruch, w ten sposób żadne dwa ruchy nie są wykonywane w tym samym czasie. Przez *pesymistyczną złożoność obliczeniową* algorytmu samstabilizującego rozumiemy maksymalną liczbę ruchów potrzebną do ustabilizowania systemu.

2 Samostabilizujące się algorytmy kolorowania grafów

Wersję optymalizacyjną problemu wierzchołkowego kolorowania grafów można sformułować w sposób następujący. Dany jest graf prosty $G = (V, E)$. Szukamy takiej funkcji $c : V \to N$(pokolorowania), że dla każdego $u, v \in V$ jeśli $\{u, v\} \in E$ to $c(v) \neq c(u)$, funkcję c spełniającą taki warunek nazywamy *pokolorowaniem legalnym*. Żądamy, by liczność zbioru przydzielonych kolorów $c(V)$ była najmniejsza z możliwych, taką liczbę oznaczamy $\chi(G)$ i nazywamy *liczbą chromatyczną* G. Problem ten jest klasycznym w teorii grafów i doczekał się bogatej literatury i wielu odmian [8]. Pokolorowanie $c : V \to N$ nazywamy pokolorowaniem Grundy'ego, jeśli dla każdego wierzchołka v mamy: jeśli $N \ni i < c(v)$, to $\exists_{u \in N(v)} c(u) = i$. W dalszym ciągu będziemy posługiwać się następującymi oznaczeniami. Niech $n = |V|$ ozna-

cza liczbę wierzchołków w grafie, a $m = |E|$ liczbę krawędzi. Ponadto przez $\deg(v)$ oznaczymy stopień wierzchołka v w grafie, a przez $\Delta(G)$ maksymalny stopień wśród wszystkich wierzchołków.

Temat kolorowania w modelach rozproszonych został podjęty niedawno między innymi w pracach Panconesiego i Srinivasana [9], Shamira i Upfala [10], Kelsena [6] i później Johanssona [5]. Pole do zastosowań praktycznych zostało pokazane w pracy [1].

Jeszcze mniej uwagi poświęcono samostabilizującym algorytmom kolorowania. Sur i Srimani w pracy [11] przedstawili algorytm dokładnego kolorowania grafów dwudzielnych, a Ghosh i Karaata w [3] podali algorytm kolorowania grafów planarnych sześcioma barwami. Natomiast w pracy [4] pokazano dwa szybkie algorytmy $\Delta(G) + 1$ kolorowania dowolnych grafów. Pierwszy z nich *Grundy coloring* jest algorytmem zachłannym stabilizuje się w czasie $O(m)$, drugi *Fast coloring* działa w czasie $O(n)$.

Sur i Srimani udowodnili, dokladność swojego algorytmu, nie podali jednak ograniczeń na czas działania W niniejszej pracy, przyjmując taki sam model obliczeń, poprawiamy wynik z roku 1993 w dwóch aspektach. Podajemy dobre ograniczenie na czas działania, oraz opisujemy algorytm dla dowolnych grafów. Rozważania zaczynamy od opisu budowy drzewa spinającego [7].

2.1 Algorytm dla drzewa spinającego

Stan każdego wierzchołka składa się z jednej zmiennej f. Na podstawie jej wartości określimy relację *„jest rodzicem"* pomiędzy wierzchołkami. Niech v będzie dowolnym wierzchołkiem w grafie G. Wybierzmy wierzchołek u w taki sposób, że $f(u) = \min_{w \in N(v)} f(w)$ tj. u jest sąsiadem v, ma minimalną wartość f spośród wszystkich sąsiadów v i jest pierwszym wierzchołkiem o żądanej własności. Jeśli $f(u) < f(v)$, to mówimy, że u jest rodzicem v i oznaczamy $p(v) = u$, w przeciwnym wypadku przyjmujemy, że v nie ma rodzica i oznaczamy $p(v) = \text{null}$. Łatwo zauważyć, że jeśli wszystkie wierzchołki, oprócz korzenia mają swojego rodzica, to wartości zmiennych f wraz z relacją „jest rodzicem" wyznaczają dla grafu $G = (V, E)$ drzewo spinające $T = (V, E')$, w którym zbiór krawędzi składa się z tych par wierzchołków, które są w relacji „jest rodzicem", $E' = \{\{v, u\} \ : \ u, v \in V \wedge u = p(v)\}$. Przyjmujemy, że dla korzenia wartość $f(r)$ jest stale równa 0. Dla pozostałych wierzchołków algorytm dany jest jedną regułą.

Algorytm 1: Drzewo spinające

F: **if** $v \neq r \wedge f(v) \leqslant \min_{u \in N(v)} f(u)$
then $f(v) = \max_{u \in N(v)} f(u) + 1$

Weźmy pod uwagę dowolnie ustalone drzewo spinające $T = (V, E_T)$ w G i krawędź $e = \{u, v\} \in E_T$, w której wierzchołek u jest bliżej korzenia niż v w drzewie T. Niech $d_T(v)$ oznacza odległość (liczbę krawędzi w najkrótszej ścieżce) od wierzchołka v do korzenia. Krawędź e nazwiemy *dobrze skierowaną*, jeśli $f(u) < f(v)$.

Twierdzenie 1 *[7] Algorytm* 1 *stabilizuje się w czasie* $O(n \operatorname{diam}(G))$ *i żaden z wierzchołków nie wykonuje więcej niż* $\operatorname{diam}(G)$ *ruchów.*

2.2 Optymalne kolorowanie grafów dwudzielnych

Zaprezentowany poniżej algorytm kolorowania opiera się na prostym pomyśle. Użyjemy funkcji określającej parzystość $A : \mathcal{N} \cup \{0\} \to \{0, 1\}$, określonej w naturalny sposób: $A(c) \equiv c \mod 2$, aby podzielić zbiór dostępnych kolorów na dwa podzbiory: *kolorów parzystych* $\{c \in \mathcal{N} \cup \{0\} : A(c) = 0\}$ i *kolorów nieparzystych* $\{c \in \mathcal{N} \cup \{0\} : A(c) = 1\}$. Funkcję parzystości można również określić inaczej, jeśli mogłoby to być użyteczne.

W dowolnym legalnym pokolorowaniu grafu każdy wierzchołek musi mieć kolor inny niż jego rodzic w dowolnie wybranym drzewie spinającym. Dlatego też proponowany algorytm przydziela najmniejszy możliwy kolor nieparzysty, jeśli rodzic jest pokolorowany parzyście, a najmniejszy możliwy kolor parzysty, jeśli rodzic jest pokolorowany kolorem nieparzystym. Dla dowolnie ustalonego wierzchołka v kolor taki oznaczmy przez $\gamma(v)$:

$$\gamma(v) = \min\{k \in N \,:\, A(k) \neq A(c(p(v))) \wedge \forall_{u \in N(v)} k \neq c(u)\}.$$

Zauważmy, że wielkość $\gamma(v)$ jest poprawnie zdefiniowana wtedy i tylko wtedy, gdy istnieje wierzchołek będący rodzicem v.

Algorytm 2: Algorytm kolorowania

`F:` **if** $v \neq r \wedge f(v) \leqslant \min_{u \in N(v)} f(u)$
then $f(v) = \max_{u \in N(v)} f(u) + 1$

`C:` **if** $p(v) \neq \text{null} \wedge c(v) \neq \gamma(v)$
then $c(v) := \gamma(v)$

Reguła `F` jest identyczna, jak w przy konstrukcji drzewa spinającego w algorytmie 1 i jej działanie nie zależy od reguły `C`, dlatego korzystając z twierdzenia 1 co najwyżej $\operatorname{diam}(G)$ ruchów zostanie wykonanych przy użyciu `F`.

Teraz oszacujemy liczbę ruchów wykonanych zgodnie z regułą `C` pomiędzy dwoma kolejnymi ruchami względem reguły `F`.

Weźmy pod uwagę dowolnie ustalony wierzchołek v. Możemy rozróżnić dwa typy ruchów wykonanych zgodnie z regułą `C`, mianowicie ruchy zmieniające parzystość zmiennej $c(v)$, które nazwiemy *ruchami przełączającymi* oraz pozostałe, które nazwiemy *ruchami nieprzełączającymi*.

Lemat 2 *Liczba ruchów przełączających wykonanych pomiędzy dwoma dowolnymi ruchami wykonanymi zgodnie z regułą `C` nie przekracza n^2.*

Dowód: Niech $T = (V, E')$ będzie lasem w G indukowanym przez relację p w następujący sposób: $u, v \in E' \Leftrightarrow u = p(v)$ lub $v = p(u)$. Zauważmy, że w każdej składowej grafu T istnieje dokładnie jeden wierzchołek u taki, że $p(u) = \text{null}$. Rozpatrzmy dowolny wierzchołek v w T. Niech r_v będzie jedynym takim wierzchołkiem, że $p(r_v) = \text{null}$ oraz zarówno v jak i r_v należą do tej samej składowej T. Poprzez $[r_v, v]_T$ oznaczmy jedyną ścieżkę z r_v do v w T, $[r_v, v] = (r_v = v_0, v_1, \ldots, v_k = v)$.

Rozpatrzmy następującą funkcję:

$$S_T(v) = \begin{cases} S_T(p(v)) + 1, & p(v) \neq \text{null} \wedge A(c(p(v)) \neq A(c(v)) \\ S_T(p(v)), & p(v) \neq \text{null} \wedge A(c(p(v)) = A(c(v)) \\ 0, & p(v) = \text{null} \end{cases}$$

Intuicyjnie, $S_T(v)$ jest liczbą krawędzi na ścieżce $[r_v, v]_T$, których incydentne wierzchołki są pokolorowane barwami różnej parzystości. Prawdziwa jest nierówność: $0 \leqslant S_T(v) < n$. Zauważmy też, że $S_T(v)$ nie zależy od wartości $f(x)$, dla $x \notin [r_v, v]_T$. Rozważmy teraz wpływ ruchu przełączającego `C` wykonanego przez $x \in [r_v, v]_T$ w $S_T(v)$. Ponieważ r_v nie mógł wykonać ruchu `C`, więc $x \neq r_v$. Są dwie możliwości: jeśli $v = x$, to $S_T(v)$ wzrasta o 1, w przeciwnym wypadku, jeśli $v \neq x$, to $S_T(v)$ nie zmienia się lub wzrasta o 2.

W konsekwencji inicjalna wartość $S_T(v)$ różni się od wartości końcowej co najwyżej o n. A ponieważ $S_T(v)$ wzrasta za każdym razem, gdy wierzchołek v wykonuje ruch przełączający, więc ruchów tych nie może być więcej niż n^2. □

Twierdzenie 3 *Algorytm 2 stabilizuje się w $O(mn^3 \operatorname{diam}(G))$ ruchów.*

Dowód: Pokażemy teraz ograniczenie na liczbę ruchów nieprzełączających pomiędzy dwoma kolejnymi ruchami przełączającymi. Ruchy nieprzełączające możemy podzielić na dwa rodzaje: te, które zwiększają wartość $c(v)$ nazywane dalej krótko *ruchami zwiększającymi* oraz pozostałe *ruchy zmniejszające*. Zauważmy, że wykonanie przez jeden wierzchołek dwóch ruchów

zmniejszających bez wykonania w systemie ruchu przełączającego nie jest możliwe, gdyż v po wykonaniu ruchu jest pokolorowany legalnie. Tak więc możliwy jest tylko jeden ruch zwiększający i to jako pierwszy w serii ruchów nieprzełączających. Wartość $c(v)$ ustalona po tym ruchu należy do zbioru $\deg(v)$ najmniejszych wartości kolorów o danej parzystości. Wobec tego liczba następujących ruchów zmniejszających nie przekracza $\deg(v)$. Sumując po wszystkich wierzchołkach otrzymujemy, że liczba wszystkich ruchów nieprzełączających pomiędzy dwoma kolejnymi przełączającymi ruchami w systemie nie przekracza $2m$.

Ostatecznie korzystając z twierdzenia 3 i lematu 2 otrzymujemy, że algorytm wykonuje co najwyżej $2m$ ruchów nieprzełączających pomiędzy dwoma kolejnymi, spośród co najwyżej n^2 ruchów przełączających `C`, dla każdego spośród co najwyżej $n \operatorname{diam}(G)$ ruchów `F`. Po przemnożeniu uzyskanych ograniczeń uzyskujemy górne ograniczenie $O(mn^3 \operatorname{diam}(G))$ ruchów, co kończy dowód. □

Twierdzenie 3 mówi, że system się ustabilizuje. Musimy jeszcze wykazać, że otrzymany wynik odpowiada naszym oczekiwaniom.

Twierdzenie 4 *Algorytm* 2 *znajduje legalne pokolorowanie grafu* G*.*

Dowód: Przypuśćmy, że system jest stabilny, wtedy żaden wierzchołek nie jest aktywny i korzystając z fałszywości predykatu reguły `C` mamy, że każde dwa wierzchołki są pokolorowane różnobarwnie, a więc pokolorowanie całego grafu jest legalne. □

Twierdzenie 5 *Algorytm* 2 *nie używa więcej niż dwóch kolorów, jeśli* G *jest dwudzielny.*

Dowód: Przypuśćmy przeciwnie, że G jest dwudzielny, a algorytm 2 stabilizuje się używając więcej niż dwóch kolorów. Zatem musiały być użyte co najmniej dwa kolory parzyste lub nieparzyste. Bez straty ogólności wybierzmy wierzchołek v spośród tych, dla których użyto drugiego z kolei koloru parzystego c. Jeśli v nie jest aktywny, to musi istnieć jakiś sąsiad $u \in N(v)$ pokolorowany pierwszym kolorem parzystym. Wtedy jednak zarówno $v \neq p(u)$ jak i $u \neq p(v)$. Musi więc w G istnieć cykl zawierający u, v i ich wspólnego rodzica. Cykl ten ma nieparzystą długość, sprzeczność z dwudzielnością G. □

2.3 Uwagi o kolorowaniu kaktusów i grafów ogólnych

Warto zauważyć, że algorytm 2 może być zastosowany do dowolnego grafu, niekoniecznie dwudzielnego. Co więcej, prawdziwy jest następujący wniosek.

Wniosek 6 *Graf G jest kolorowany algorytmem 2 w ciągu $O(mn^3 \operatorname{diam}(G))$ ruchów przy użyciu co najwyżej 2Δ kolorów.*

Niestety jest to wynik słaby w porównaniu z innymi znanymi algorytmami dla grafów ogólnych. Przykładowo algorytm pokazany w pracy [4] używa co najwyżej $\Delta + 1$ kolorów i działa szybciej, można jednak bez trudu skonstruować graf dwudzielny, który zostanie pokolorowany tym algorytmem dokładnie $\Delta + 1$ kolorami. Oczywiście nic podobnego nie może się zdarzyć w wypadku algorytmu 2.

W ogólności algorytm 2 nie koloruje grafów trójdzielnych optymalnie. Możemy jednak sformułować dobre ograniczenie na liczbę użytych kolorów dla pewnej rodziny grafów trójdzielnych, mianowicie kaktusów.

Twierdzenie 7 *Kaktus G jest kolorowany przez algorytm 2 przy użyciu co najwyżej 4 kolorów.*

Dowód: Dowiedziono, że algorytm stabilizuje się. Wystarczy pokazać, że w przypadku kaktusów liczba użytych kolorów nie przekroczy 4. Niech wobec tego G będzie kaktusem, a T jego drzewem spinającym utworzonym w trakcie działania algorytmu. Rozpatrzmy dowolnie ustalony wierzchołek $v \in G$ nie będący korzeniem. Niech $N_A(v) = \{u \in N(v) : A(u) = A(v)\}$, oczywiście $c(v) < 2(|N_A(v)|+1)$. Przypuśćmy, że $u \in N_A(v)$, wtedy $u \neq p(v)$ i $v \neq p(u)$.

Ponadto u nie należy do ścieżki $[v, r]_T$, gdyż w przeciwnym wypadku mielibyśmy ścieżkę $v = v_0, v_1, \ldots, v_k = u$, gdzie $v_i = p(v_{i-1})$ dla $i = 1, 2, \ldots, k$ i $f(v_i) < f(v_{i-1}) < \ldots < f(v_0)$, co przeczy, że $p(v)$ jest sąsiadem wierzchołka v takiego, że $f(p(v)) = \min\{f(w) : w \in N(v)\}$. Podobnie v nie należy do ścieżki $[u, r]_T$. Mamy więc cykl w G zawierający u, v, $p(v)$, ale G jest kaktusem, gdzie krawędź $\{v, p(v)\}$ może występować tylko w jednym cyklu, wobec tego w $N_A(v)$ jest co najwyżej jeden wierzchołek; a więc $c(v) < 4$. Ponieważ wartość c dla korzenia wynosi 0, to dowód został zakończony. □

Literatura

[1] Battiti R., Bertossi A. A., Bonuccelli M. A.: Assigning codes in wireless networks. Wireless Networks 5, 1999, 195–209.

[2] Dijkstra E. W.: Self-stabilizing systems in spite of distributed control. Communications of the ACM 17, 1974, 643–644.

[3] Ghosh S. , Karaata M. H.: A self-stabilizing algorithm for coloring planar graphs. Distributed Computing 7, 1993, 55–59.

[4] Hedetniemi S. T., Jacobs D. P., Srimani P. K.: Linear time self-stabilizing colorings. Information Processing Letters 87, 2003, 251–255.

[5] Johansson Ö.: Simple distributed $\Delta + 1$ — coloring of graphs. Information Processing Letters 70, 1999, 229–232.

[6] Kelsen P.: Neighborhood graphs and distributed (Δ+1)-coloring. Proc. SWAT LNCS 1097, 1996, 223–233.

[7] Kosowski A., Kuszner Ł.: A self-stabilizing algorithm for finding a spanning tree in a polynomial number of moves. Proc. PPAM LNCS 3911, 2006, 75–82.

[8] Kubale M. i inni: Graph Colorings, Contemporary Mathematics 352, AMS, 2004.

[9] Panconesi A., Srinivasan A.: On the complexity of distributed network decomposition. Journal of Algorithms 20, 1996, 356–374.

[10] Shamir E., Upfal E.: Sequential and distributed graph coloring algorithms with performance analysis in random graph spaces. Journal of Algorithms 5, 1984, 488–501.

[11] Sur S., Srimani P. K.: A self-stabilizing algorithm for coloring bipartite graphs. Information Sciences 69, 1993, 219–227.

Abstract

In the paper a distributed self-stabilizing algorithm for graph coloring is given. We consider a connected system of autonomous asynchronous nodes, each of which has only local information about the system. Regardless of the initial state, the system must achieve a desirable global state by executing a set of rules assigned to each node. Our method based on spanning tree construction is applied to give the first (to our knowledge) self-stabilizing algorithms working in a polynomial number of moves, which color bipartite graphs with exactly two colors. The complexity and performance characteristics of the presented algorithms are discussed. The number of moves performed by the algorithm is bounded by $O(mn^3 \operatorname{diam}(G))$, where $\operatorname{diam}(G) < n$ is the distance between the furthest pair of nodes of the system. Finally, we analyze the performance of the presented algorithm for arbitrary and cacti graphs, stating that they use not more than 2Δ colors in general case and not more than 4 colors in the case of cacti.