ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2006
Seria: AUTOMATYKA z. 143 Nr kol. 1726

Paweł OBSZARSKI, Konrad PIWAKOWSKI
Politechnika Gdańska

ZASTOSOWANIE ALGORYTMÓW ROJOWYCH DO KOLOROWANIA GRAFÓW

Streszczenie. Przedstawiamy sposób adaptacji heurystycznej metody przeszukiwania PSO (ang. Particle Swarm Optimization) do znajdowania suboptymalnych pokolorowań wierzchołkowych grafów prostych. Prezentujemy sposób przeprowadzenia eksperymentów obliczeniowych oraz ich wyniki.

APPLYING PSO PARADIGM TO GRAPH COLORING

Summary. Adaptation of the Particle Swarm Optimization method for obtaining suboptimal vertex colorings of graphs is proposed. We present details of performed computational experiments and their results.

1. Wierzchołkowe kolorowanie grafów – algorytmy sekwencyjne

W niniejszej pracy rozważamy wyłącznie *grafy proste*, tzn. niekierowane, bez pętli i krawędzi równoległych. Dla danego grafu $G = (V(G), E(G))$ *pokolorowaniem (wierzchołkowym)* nazywamy dowolną funkcję $f : V(G) \to N$ taką, że dla każdej krawędzi $vw \in E(G)$ zachodzi $f(v) \neq f(w)$. Pokolorowanie nazywamy *k-pokolorowaniem*, gdy $|f(V(G))| \leq k$. Najmniejszą liczbę $k \in N$, dla której istnieje k-pokolorowanie grafu G nazywamy liczbą chromatyczną i oznaczamy przez $\chi(G)$. Wyznaczanie dokładnej wartości liczby chromatycznej oraz znajdowanie optymalnego pokolorowania (czyli $\chi(G)$-pokolorowania) są problemami NP-trudnymi [1]. Wobec braku efektywnych metod znajdowania rozwiązań dokładnych, w praktyce często stosowane są algorytmy przybliżone, wśród których istotne miejsce zajmują algorytmy sekwencyjne, realizujące następujący schemat:

Krok 1. Ustal kolejność wierzchołków $v_1, v_2, ..., v_n$.
Krok 2. for $i := 1$ **to** n **do**
Przydziel wierzchołkowi v_i najmniejszy możliwy kolor (nie zablokowany przez pokolorowanych sąsiadów).

Krok 2 jest w tym schemacie określony jednoznacznie. Krok 1 natomiast pozostawia natomiast wielką dowolność. Dla dowolnego grafu G można zawsze znaleźć taką kolejność wierzchołków, że realizacja kroku 2 prowadzi do pokolorowania optymatnego. W tym celu wystarczy wziąć dowolne pokolorowanie optymalne f i posorto-

wać wierzchołki w kolejności niemalejących kolorów $f(v_i) \leq f(v_j)$, dla $i < j$. Dla tak wybranej kolejności pokolorowanie g, wygenerowane w kroku 2 spełnia $g(v_i) \leq f(v_i)$, dla $1 \leq i \leq n$, a zatem jest także pokolorowaniem optymalnym. Algorytmy sekwencyjne różnią się między sobą sposobem realizacji kroku 1. Jedna z najbardziej popularnych heurystyk, *metoda LF* (ang. Largest First) [6], porządkuje wierzchołki w kolejności nierosnących stopni $\deg(v_i) \geq \deg(v_j)$, dla $i<j$, gdzie $\deg(v) = |\{w \in V : vw \in E\}|$. Jest ona zgodna z intuicją, że lepiej kolorować wierzchołki o dużym stopniu wcześniej, aby, gdy jest już wiele wierzchołków pokolorowanych, które tym samym wprowadzają ograniczenia przyznawania kolorów nowym wierzchołkom, pozostawały do pokolorowania wierzchołki o możliwie jak najmniejszym stopniu. Ta sama intuicja tkwi w *algorytmie SL* (ang. Smallest Last) [5], gdzie kolejność wierzchołków jest ustalana w n krokach w ten sposób, że w i-tym kroku wierzchołek o najmniejszym stopniu jest umieszczany na $n - i + 1$ pozycji sekwencji i usuwany wraz z incydentnymi krawędziami z grafu. Metody te są efektywne, a dla niektórych klas grafów uzyskują rozwiązania optymalne lub bliskie optymalnym. Na przykład algorytm SL koloruje optymalnie drzewa i nie zużywa nigdy więcej niż 6 kolorów do kolorowania grafów planarnych. Niestety dla każdej z obu tych metod można wskazać przykłady grafów, dla których uzyskanie optymalnych pokolorowań jest niemożliwe [3,4]. Wolny od tej wady jest *losowy algorytm sekwencyjny*, gdzie w pierwszym kroku wybór sekwencji odbywa się bez żadnej heurystyki – zupełnie losowo. W tym przypadku dowolny graf ma szanse zostać pokolorowany optymalnie, ale średnio liczba wykorzystywanych kolorów przez ten algorytm jest większa niż przez algorytmy LF i SL. W dalszej części przedstawimy nową metodę realizacji wyboru sekwencji wierzchołków w algorytmie sekwencyjnym opartą na algorytmie rojowym.

2. Paradygmat przeszukiwania PSO

Idea poszukiwania ekstremum nieliniowych ciągłych funkcji wielu zmiennych, inspirowana zachowaniem się kluczy ptaków, ławic ryb i rojów owadów PSO (ang. Particle Swarm Optimization), została zaprezentowana po raz pierwszy w 1995 roku przez Kennedy'ego i Eberhart'a [2]. W tym podejściu pewna skończona liczba k osobników ze zbioru osobników P przemieszcza się w przestrzeni X, w której każdemu punktowi $x \in X$ przyporządkowana jest określona wartość funkcji celu $f(x)$, której minimum pragniemy zlokalizować. Stan każdego osobnika $p \in P$ jest określony przez aktualne położenie $x_p \in X$ i aktualną prędkość v_p. Oprócz tego, każdy osobnik pamięta odwiedzony przez siebie punkt $b_p \in X$, w którym wartość funkcji celu f jest najmniejsza, spośród wszystkich punktów przestrzeni odwiedzonych dotąd przez tego osobnika. Po losowym nadaniu inicjalnych wartości prędkości i położenia każdemu z osobników, realizowana jest pętla złożona z trzech kolejnych kroków:

Krok 1. Wyznaczany jest osobnik q, dla którego $b_q = \min\{b_p : p \in P\}$.

Krok 2. Wyznaczana jest nowa wartość prędkości dla każdego osobnika p:

$v_p' = c_v v_p + \text{rand} \cdot c_a(b_p - x_p) + \text{rand} \cdot c_b((b_q - x_p)$, gdzie rand jest losową wartością z przedziału (0, 1).

Krok 3. Wyznaczana jest nowa wartość położenia dla każdego osobnika p:

$x_p' = x_p + v_p.'$

W kroku 1 wyznaczany jest osobnik q, który odwiedził kiedykolwiek punkt przestrzeni X, w którym wartość funkcji celu f jest najmniejsza spośród wszystkich punktów przestrzeni odwiedzonych dotąd przez wszystkich osobników, tzn. $b_q = \min\{b_p : p \in P\}$. W kroku 2 każdy osobnik p zmienia swój wektor prędkości poprzez wyznaczenie kombinacji liniowej wektorów aktualnej swojej prędkości v_p: wektora prowadzącego z aktualnego położenia x_p do "najlepszego" odwiedzonego w przeszłości przez tego osobnika miejsca b_p, oraz wektora prowadzącego z aktualnego położenia x_p do "najlepszego" miejsca przestrzeni X odwiedzonego dotąd przez jakiegokolwiek osobnika ze zbioru P. Współczynniki c_v, c_a oraz c_b są stałymi, określającymi wagę wpływu tych poszczególnych czynników zaburzaną przez wprowadzenie losowej funkcji *rand* (chwilowo osobnik decyduje się podążać za swoim najlepszym wspomnieniem, decyduje się podążać w kierunku tego, który zgłasza, że odkrył coś lepszego albo po prostu podąża w swoim aktualnym kierunku).

Możliwość skutecznego wykorzystania tej metody do rozwiązywania problemów optymalizacyjnych, gdzie przestrzeń stanów jest dyskretna i funkcja stanu jest nieciągła, wymaga zdefiniowania takich operatorów dodawania wektorów położenia i prędkości, mnożenia wektora prędkości przez skalar oraz wyznaczania wektora prędkości z różnicy dwóch wektorów położenia, aby wykonanie przemieszczenia z pozycji o większej wartości funkcji celu w kierunku pozycji o mniejszej wartości funkcji celu statystycznie zwiększało szanse na osiągniecie pozycji o mniejszej wartości funkcji celu. Innymi słowy, wykonanie kroku w kierunku "lepszej" pozycji częściej powoduje poprawę aktualnej pozycji niż jej pogorszenie.

Propozycję adaptacji paradygmatu PSO do poszukiwania pokolorowań wierzchołkowych grafów prostych przedstawiamy w następnej sekcji.

3. Adaptacja algorytmu PSO do poszukiwania pokolorowań wierzchołkowych

Dla danego grafu G, gdzie $|V(G)| = n$, zbiór wszystkich sekwencji wierzchołków $V(G)$ utożsamiamy z przestrzenią stanów X, gdzie $X \subseteq R^n$. Element x_i wektora $x = (x_1, x_2, ..., x_n)$ określa pozycję wierzchołka v_i w odpowiadającej mu sekwencji. I tak np. sekwencji wierzchołków v_3, v_1, v_2, v_4 odpowiada stan (2, 3, 1, 4).

Sumowanie wektorów i mnożenie wektorów przez skalar jest określone jak w zwykłej przestrzeni euklidesowej R^n. Elementy przestrzeni X są traktowane jak elementy R^n, a po wykonaniu operacji algebraicznych uzyskany wynik $y \in R^n$ jest normalizowany do elementu $x \in X$ zgodnie z formułą $x_i = |\{y_j : y_j \le y_i, 1 \le j \le n\}|$.

Rozważmy następujący przykład. Dla grafu o czterech wierzchołkach $V = \{v_1, v_2, v_3, v_4\}$ dane są dwie sekwencje wierzchołków $x = (v_2, v_4, v_3, v_1)$ oraz $y = (v_2, v_1, v_3, v_4)$. Przeanalizujmy wykonanie operacji $z = 1.5x - y$. Stany odpowiadające x i y odpowiednio przyjmują wartości (4, 1, 3, 2) oraz (2, 1, 3, 4). Po wykonaniu operacji algebraicznych w R^n otrzymujemy w wyniku wektor (4, 0.5, 1.5, –1), który po normalizacji przyjmuje wartość (4, 2, 3, 1), co dopowiada sekwencji $z = (v_4, v_2, v_3, v_1)$.

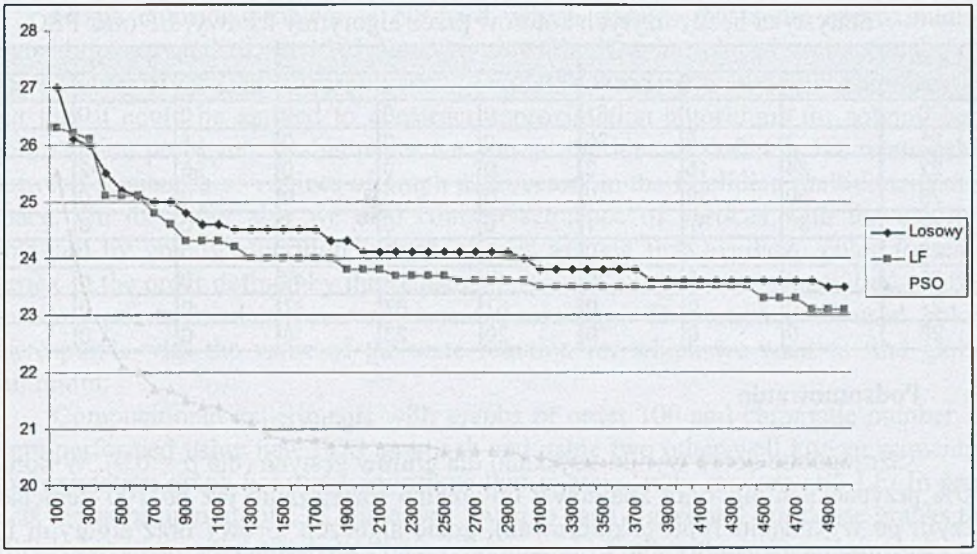
Przedstawiony powyżej algorytm nie wyznacza pokolorowania, lecz jedynie sekwencję wierzchołków, która służy do wyznaczenia pokolorowania zgodnie z drugim krokiem algorytmu sekwencyjnego omówionego w pierwszej sekcji. Dla danego stanu x wartość funkcji celu $f(x)$ jest równa liczbie kolorów wykorzystanych przez algorytm zachłanny dla sekwencji wierzchołków odpowiadającej stanowi x.

Wyniki eksperymentów obliczeniowych oraz porównanie skuteczności nowego podejścia z algorytmami losowym oraz LF przedstawiamy w następnej sekcji.

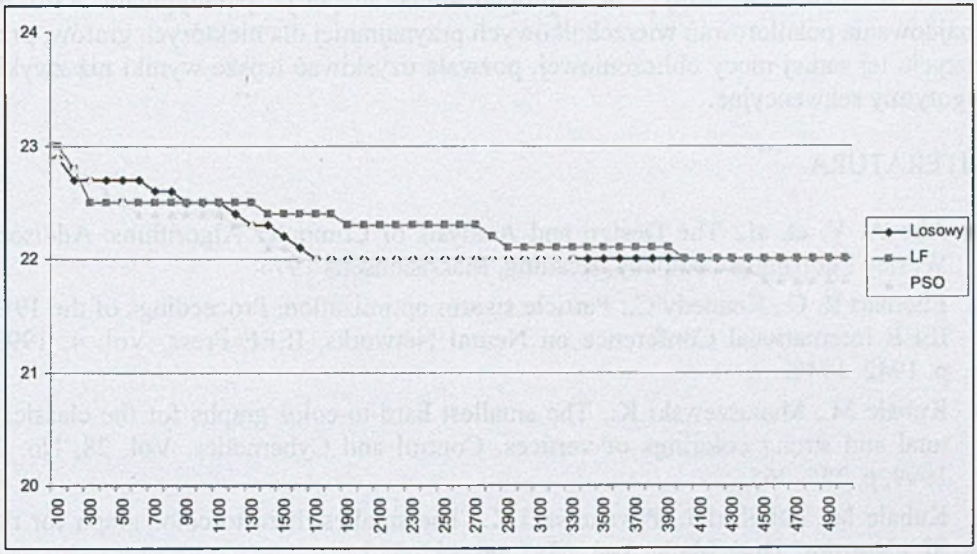
4. Wyniki eksperymentów obliczeniowych

Przeprowadzono eksperymenty na grafach losowych o 100 wierzchołkach, liczbie chromatycznej $\chi = 20$ i różnej gęstości krawędzi. Grafy były generowane w ten sposób, że zbiór wierzchołków dzielono na 20 równolicznych zbiorów niezależnych $V = V_1 \cup V_2 \cup \ldots \cup V_{20}$. Z każdego tych zbiorów wybierano jednego reprezentanta $u_i \in V_i$ i każde dwa wierzchołki u_i, u_j, dla $1 \leq i < j \leq 20$ łączono krawędzią, co powodowało powstanie podgrafu pełnego K_{20}. Dla każdej pozostałej pary wierzchołków $v \in V_i$, $w \in V_j$, $i \neq j$, dołączano krawędź vw z prawdopodobieństwem p, której wartość wpływała na gęstość generowanych grafów. Ponieważ tak wygenerowany graf G zawiera podgraf pełny K_{20} więc $\chi(G) \geq 20$. Z drugiej strony zawiera 20 zbiorów niezależnych, więc $\chi(G) \leq 20$. Dokładna znajomość liczby chromatycznej pozwoliła precyzyjnie wyznaczyć błąd popełniany przez testowane algorytmy przybliżone.

Eksperymentalnie dopasowano następujące parametry: $c_v = 0.7$, $c_a = 1.8$, $c_a = 1.9$ oraz przyjęto stałą liczbę 10 osobników w roju. W ramach każdej iteracji konieczne więc było dziesięciokrotne wykonanie kolorowania zachłannego. Przeprowadzono eksperymenty dla grafów o różnych gęstościach, gdzie parametr p przyjmował odpowiednio wartości 0.6, 0.75 oraz 0.9. Eksperyment przerywano po 500. iteracji, czyli po wykonaniu 5000 pokolorowań. Na tych samych grafach przeprowadzano analogicznie po 5000 pokolorowań metodą losową oraz metodą LF. Rysunki 1 i 2 przedstawiają wykres najlepszego uzyskanego rezultatu przez każdą z metod w funkcji liczby wykonanych iteracji (wyniki uśredniono powtarzając eksperyment dziesięciokrotnie). Przeprowadzono także drugi eksperyment, polegający na stukrotnym powtórzeniu wyznaczania możliwie jak najlepszego pokolorowania tego samego grafu każdą z trzech następujących metod: (a) wybór najlepszego spośród 1000 pokolorowań metodą losową, (b) wybór najlepszego spośród 1000 pokolorowań metodą LF oraz (c) zastosowanie algorytmu rojowego, zatrzymanego po 100 iteracjach (czyli także po wykonaniu 1000 pokolorowań). Wyniki drugiego eksperymentu zostały przedstawione w tabeli 1, gdzie komórka w wierszu i i kolumnie j zawiera liczbę przypadków uzyskania liczby kolorów określonej w pierwszej kolumnie i tym samym wierszu za pomocą metody i dla parametru gęstości grafu określonego w nagłówku kolumny. Na przykład dla $p = 0.9$ algorytm POS uzyskał 61 razy pokolorowanie optymalne, 35 razy pokolorowanie 21-kolorowe, 3 razy 22-kolorowe i 1 raz pokolorowanie 24-kolorowe.



Rys.1. Najlepsze uzyskane pokolorowanie w funkcji liczby wykonanych iteracji przez algorytmy losowy, LF oraz PSO dla parametru gęstości grafu $p = 0.9$. Skala na osi pionowej wskazuje liczbę użytych kolorów, a na osi poziomej liczbę wykonanych pokolorowań od momentu rozpoczęcia wykonywania algorytmu



Rys.2. Najlepsze uzyskane pokolorowanie w funkcji liczby wykonanych iteracji przez algorytmy losowy, LF oraz PSO dla parametru gęstości grafu $p = 0.6$. Skala na osi pionowej wskazuje liczbę użytych kolorów a na osi poziomej liczbę wykonanych pokolorowań od momentu rozpoczęcia wykonywania algorytmu

Tabela 1

Statystyka liczby użytych kolorów przez algorytmy losowy, LF oraz PSO

Liczba kolorów	p = 0.9			p = 0.75			p = 0.6		
	Losowy	LF	PSO	Losowy	LF	PSO	Losowy	LF	PSO
20	1	0	61	0	0	0	0	0	0
21	3	5	35	0	0	0	5	7	23
22	24	15	3	0	0	0	95	92	77
23	30	36	0	0	0	0	0	0	0
24	36	37	1	0	0	1	0	0	0
25	6	7	0	1	1	12	0	0	0
26	0	0	0	12	9	47	0	0	0
27	0	0	0	67	67	37	0	0	0
28	0	0	0	20	23	3	0	0	0

5. Podsumowanie

Szczególnie dobre wyniki uzyskano dla grafów gęstych (dla $p = 0.9$). W ponad 60% przypadkach algorytm znajdował optymalne rozwiązanie już po 100 iteracjach (czyli, po wykonaniu 1000 pokolorowań), gdzie algorytm losowy oraz algorytm LF znajdowały pokolorowania wykorzystujące najczęściej 2, 3 lub 4 kolory więcej. Różnicę tą widać wyraźnie na wykresie na Rysunku 1 oraz porównując statystyki liczby użytych kolorów z pierwszych czterech kolumn tabeli 4.1. Dla mniejszych wartości parametru p przewaga algorytmu rojowego nad pozostałymi dwoma nie jest już tak spektakularna (patrz rys. 2 oraz prawa część tabeli 4.1).

Uzyskane wyniki wskazują, że zaproponowana adaptacja paradygmatu PSO do znajdowania pokolorowań wierzchołkowych przynajmniej dla niektórych grafów, przy zużyciu tej samej mocy obliczeniowej, pozwala uzyskiwać lepsze wyniki niż zwykłe algorytmy sekwencyjne.

LITERATURA

1. Aho A. V. et. al.: The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Company, Reading, Massachusetts 1974.
2. Eberhart R. C., Kennedy C.: Particle swarm optimization. Proceedings of the 1995 IEEE International Conference on Neural Networks, IEEE Press, Vol. 4, 1995, p. 1942–1948.
3. Kubale M., Manuszewski K.: The smallest hard-to-color graphs for the classical, total and strong colorings of vertices. Control and Cybernetics, Vol. 28, No. 2, 1999, p. 355–365.
4. Kubale M., Pakulski J., Piwakowski K.: The smallest hard-to-color graph for the SL algorithm. Disc. Math., 164, 1997, pp. 197–212.
5. Matula D. W. et. al.: Graph Coloring Algorithms, Graph Theory and Computing. Academic Press, New York 1972, p. 109–122.
6. Powell M. B., Welsh P. J.: An upper bound for the chromatic number of graph and its applications to timetabling problem. Comput. J. 10, 1967, p. 85–86.

Recenzent: Prof. dr hab.inż. Adam Janiak

Abstract

Graph coloring problem is NP-hard, which implies that some approximation algorithms are studied. In 1995 Kennedy and Eberhart introduced a concept for the optimization of nonlinear functions using particle swarm methodology. Later it turned out that it could be applied to construct approximation algorithms for solving hard optimization problems. In this paper we present the idea of defining 1-1 relationship between sequences of vertices of graph and vectors in the Euclidian multidimensional space. On the other side we also connect sequence of vertices with the coloring obtained by greedy algorithm (which always assigns first available color for each vertex in the order defined by the sequence). In such a way we have specified relation between points in the space and vertex colorings. The number of used colors corresponds with the value of the state function for which we want to find global minimum.

Computational experiments with graphs of order 100 and chromatic number 20 were performed using new POS approach and using two other well known sequential approximation graph coloring algorithms that is completely random and LF. In each tests the same amount of CPU time was given to each algorithm. For dense graphs the PSO behaved significantly better than random and LF algorithms. Detailed results are presented in a table and charts.