Projekt – Izolacja

Tomasz Zawadzki

12 lipca 2020

1 Streszczenie

Wczytana plansza jest konwertowana na graf G^L reprezentowany jako listy sąsiedztwa, w którym wyszukiwany jest zbiór niezależny o liczności K przy użyciu algorytmu rekurencyjnego z optymalizacjami dla wierzchołków stopnia 0, 1 oraz 2.

Każda z prób zrealizowania optymalizacji dla „zwijalnego" (ang. *foldable*) wierzchołka stopnia 3 zakończyła się niepowodzeniem najprawdopodobniej z uwagi na błędy implementacyjne związane z interpretacją wybierania i odrzucania nowych wierzchołków.

Podział grafu G^L na składowe C_i umożliwia znajdowanie zbiorów niezależnych osobno dla każdej składowej przy jednoczesnym zmniejszeniu złożoności obliczeniowej

$$O^*(\varphi^n + \varphi^m) < O^*(\varphi^{n+m}),$$

lecz ta wersja programu działa znacznie wolniej niż wyszukiwanie zbioru niezależnego dla całego grafu G^L, ponieważ (przy najprostszej strategii) wymaga obliczenia $\alpha(C_i)$ dla $n-1$ „najłatwiejszych" komponentów, a także z uwagi na brak informacji, ile wierzchołków należy wybrać z poszczególnych składowych, w przypadku gdy $K < \sum_i \alpha(C_i)$.

Próba zredukowania instancji problemu Independent-Set do Integer-Linear-Programming oraz wykorzystania solvera ILP z pakietu GLPK do znalezienia rozwiązania zakończyła się powodzeniem, ale napotkałem spore trudności z umieszczeniem kodu źródłowego solvera w jednym pliku `*.cpp`.

2 Algorytm

2.1 Wczytanie planszy

Plansza jest wczytywana ze standardowego wejścia do dwuwymiarowej tablicy znaków.

2.2 Obsługa przypadków trywialnych (opcjonalne)

Dla $K = 0$ program natychmiast kończy działanie.
Dla $L = 0$ program wypisuje K pierwszych niepustych komórek.

2.3 Utworzenie grafu G^L

Niech $G = (V, E)$ będzie grafem nieskierowanym reprezentującym wczytaną planszę, gdzie pola planszy niebędące pustymi komórkami (.) są reprezentowane jako wierzchołki grafu G, natomiast możliwość przejścia pomiędzy sąsiadującymi komórkami jest reprezentowana jako krawędź.

Niech G^L oznacza L-tą potęgę grafu bez pętli własnych

$$G^L = (V(G), \{\{u, w\} \mid 1 \leqslant \text{distance}_G(u, w) \leqslant L\}) \tag{1}$$

W szczególności $G^0 = (V(G), \varnothing)$ oraz $G^1 = G$. Na podstawie wczytanej planszy konstruowany jest graf G^L przechowywany w postaci list sąsiedztwa.

2.4 Uwzględnienie wysp o rozmiarze 1 (opcjonalne)

Po utworzeniu grafu G^L można dodać do zbioru niezależnego wszystkie wierzchołki izolowane („wyspy" o rozmiarze 1) oraz usunąć je z grafu.

2.5 Podział grafu G^L na składowe C_i (opcjonalne)

Za pomocą algorytmu BFS przeprowadzany jest podział grafu G^L na składowe $\{C_i\}_{i=1}^n$. Niech $\mathrm{IS}_k(G)$ oznacza dowolny zbiór niezależny rozmiaru k w grafie G. Niech $\mathrm{MIS}(G)$ oznacza dowolny maksymalny zbiór niezależny grafu G, natomiast $\alpha(G)$ – jego liczność.

W przypadku gdy $K = \alpha(G^L) = \sum_{i=1}^{n} \alpha(C_i)$, jedynym rozwiązaniem problemu jest suma mnogościowa zbiorów niezależnych każdej składowej C_i grafu G^L, $\mathrm{MIS}(G^L) = \bigcup_i \mathrm{MIS}(C_i)$.

W przypadku gdy $K < \alpha(G^L)$, nie posiadamy informacji o wymaganych licznościach zbiorów niezależnych poszczególnych składowych. Obliczenie $\alpha(C_i)$ wymaga sprawdzenia wszystkich rozgałęzień drzewa, co znacznie zwiększa wykorzystanie pamięci oraz czas działania programu. W szczególności dla $G = C_1 + C_2$ oraz $K = \alpha(G) - 1$, istnieją dwa istotnie różne rozwiązania:

$$\mathrm{MIS}(C_1) \cup \mathrm{IS}_{\alpha(C_2)-1}(C_2) \text{ oraz } \mathrm{IS}_{\alpha(C_1)-1}(C_1) \cup \mathrm{MIS}(C_2). \tag{2}$$

W przypadku większej liczby składowych oraz różnicy $\alpha(G^L) - K$, praktycznie niemożliwe jest określenie optymalnych wartości $\{k_i\}_{i=1}^n$ rozwiązania $\bigcup_i \mathrm{IS}_{k_i}(C_i)$.

2.5.1 Analogia do sudoku

Trafną analogią jest rozwiązywanie sudoku – intuicyjnie łatwiej jest uzupełnić dwa sudoku z maksymalnie dwoma konfliktami (np. przy użyciu prostego algorytmu zachłannego) niż uzupełnić jedno sudoku z maksymalnie czterema konfliktami, a drugie rozwiązać prawidłowo.

2.5.2 Heurystyka

Jedną ze strategii może być heurystyka polegająca na posortowaniu ciągu składowych zgodnie z pewną metryką (np. według liczby wierzchołków albo stosunku $|E|/|V|$) określającą trudność znalezienia maksymalnego zbioru niezależnego tej składowej, w kolejności od najłatwiejszej C_1 do najtrudniejszej C_n. Wówczas rozwiązaniem jest

$$\mathrm{MIS}(C_1) \cup \mathrm{MIS}(C_2) \cup \ldots \cup \mathrm{MIS}(C_{n-1}) \cup \mathrm{IS}_{k_n}(C_n), \tag{3}$$

gdzie

$$k_n = K - \sum_{i=1}^{n-1} \alpha(C_i) \leqslant K, \tag{4}$$

a równość zachodzi tylko dla $K = \alpha(G^L)$, co wyklucza konieczność potencjalnie czasochłonnego znajdowania $\mathrm{MIS}(C_n)$ w przypadku, gdy nierówność jest słaba. Niestety, program realizujący te założenia działa znacznie wolniej niż podstawowa wersja wyszukująca $\mathrm{IS}_K(G^L)$.

2.6 Znalezienie zbioru niezależnego o liczności K

Problem Independent-Set jest NP-zupełny, zatem nie istnieje algorytm dokładny działający w czasie wielomianowym (o ile P $\neq$ NP). Zbiór niezależny jest wyszukiwany za pomocą standardowego algorytmu rekurencyjnego wykorzystującego zasady rozgałęziania i redukcji.

Niech $N(v)$ oznacza otwarte sąsiedztwo wierzchołka v,

$$N(v) = \{w \mid \{v, w\} \in E\}, \tag{5}$$

natomiast $N[v]$ oznacza domknięte sąsiedztwo wierzchołka v,

$$N[v] = N(v) \cup \{v\}. \tag{6}$$

2.6.1 Przypadki bazowe rekurencji

Liczność maksymalnego zbioru niezależnego w grafie pustym G wynosi zero.

$$\alpha(G) = 0 \tag{7}$$

Zbiór niezależny rozmiaru 0 jest zbiorem pustym.

$$IS_0(G) = \varnothing \tag{8}$$

Jeśli G jest grafem pustym, to nie istnieje zbiór niezależny rozmiaru $k > 0$.

$$IS_k(G) \text{ nie istnieje} \tag{9}$$

2.6.2 Redukcja dla wierzchołków izolowanych

Jeśli v jest wierzchołkiem izolowanym (czyli stopnia 0), możemy dodać ten wierzchołek do zbioru niezależnego oraz usunąć go z grafu.

$$\mathrm{IS}_k(G) = \{v\} \cup \mathrm{IS}_{k-1}(G - \{v\}) \tag{10}$$

$$\alpha(G) = 1 + \alpha(G - \{v\}) \tag{11}$$

2.6.3 Redukcja dla wierzchołków stopnia 1

Jeśli v jest wierzchołkiem stopnia 1, spełniona jest zależność

$$\alpha(G) = 1 + \alpha(G - N[v]). \tag{12}$$

Oznacza to, że możemy uwzględnić ten wierzchołek w zbiorze niezależnym oraz usunąć z grafu jego sąsiedztwo domknięte.

$$\mathrm{IS}_k(G) = \{v\} \cup \mathrm{IS}_{k-1}(G - N[v]) \tag{13}$$

2.6.4 Redukcja dla wierzchołków stopnia 2

Niech v będzie wierzchołkiem stopnia 2. Oznaczmy jego sąsiadów jako w_1 oraz w_2.

Jeśli w grafie istnieje krawędź $\{w_1, w_2\}$, to uwzględniamy wierzchołek v w rozwiązaniu, natomiast z grafu usuwamy jego sąsiedztwo $N[v] = \{v, w_1, w_2\}$.

Jeśli wierzchołki w_1 i w_2 nie są połączone krawędzią, wykonujemy operację *fold* polegającą na dodaniu do grafu nowego wierzchołka z wraz z krawędziami do $N(w_1) \cup N(w_2) - \{v\}$ oraz usunięciu z grafu $N[v]$. Otrzymany graf oznaczmy jako G'. Wówczas

$$\alpha(G) = 1 + \alpha(G'). \tag{14}$$

Jeśli wierzchołek z znajdzie się w rozwiązaniu, zamieniamy go na w_1 oraz w_2, a jeśli zostanie odrzucony, to uwzględniamy w rozwiązaniu wierzchołek v.

2.6.5 Reguły rozgałęzienia dla pozostałych wierzchołków

Po zaaplikowaniu wyżej wymienionych redukcji aż do wyczerpania otrzymamy graf, którego wszystkie wierzchołki mają stopień co najmniej 3. Dla dowolnego wierzchołka v grafu G spełniona jest zależność rekurencyjna

$$\alpha(G) = \max\{1 + \alpha(G - N[v]),\ \alpha(G - \{v\})\}. \tag{15}$$

Spośród wierzchołków grafu wybieramy wierzchołek v o najmniejszym stopniu. Wówczas

$$\mathrm{IS}_k(G) = \arg\max_{|\cdot|}\{\{v\} + \mathrm{IS}_{k-1}(G - N[v]),\ \mathrm{IS}_k(G - \{v\})\}. \tag{16}$$

2.7 Wypisanie współrzędnych wybranych pól

Współrzędne pól ze znalezionego zbioru niezależnego są wypisywane na standardowe wyjście.

Bibliografia

[1] "Exact Algorithm for Independent Set"
`http://cs.lth.se/fileadmin/cs/EDAN55/independentset.pdf`

[2] "Exact Algorithms for Maximum Independent Set"
`http://people.idsia.ch/~grandoni/Pubblicazioni/G14ea.pdf`

[3] `http://people.idsia.ch/~grandoni/Pubblicazioni/FGK06soda.pdf`