Roman Moskalenko
295136

AAL Zadanie 15 – Integracja

Treść zadania

Pracownicy wielkiego zakładu produkcyjnego wyjeżdżają na wyjazd integracyjny w kiku turnusach. Dział HR chce, aby jego trakcie pracownicy lepiej się poznali, dlatego w jednym turnusie będą uczestniczyły osoby, które się wcześniej nie znały. Mając dane informacje, mówiące o tym które osoby się znają, zaproponować algorytm, który wyznaczy minimalną liczbę potrzebnych turnusów, oraz dokona przydziału pracowników.

Przyjęte założenia

- Sytuacja w której jeden pracownik zna drugiego, ale drugi nie zna pierwszego nie jest rozpatrywana.
- Zakłada się, że dział HR nie ma zamiaru znajomić pracowników by każdy znał każdego. Natomiast każdy pracownik musi wziąć udział w turnusie dokładnie jeden raz, chyba że są tacy którzy znają wszystkich innych pracowników, wtedy nie powinni uczęsticzyć w turnusie.
- Każdy turnus musi liczyć co najmniej 2 pracowników. Jeśli nie jest to możliwe do spełnienia, stwierdzamy, że nie da się dokonać przydziału.

Analiza problemu

- Problem ten polega na optymalnym kolorowaniu grafu, tzn. kiedy zużywa się minimalnej liczby kolorów dla rozwiązania problemu.
- Pracownicy reprezentowani są przez wierzchołki grafu nieskierowanego, krawędzie wskazują na znajomość pomiędzy dwoma pracownikami. Liczba turnusów odpowiada liczbie chromatycznej grafu, przydział pracowników – właśnie pokolorowanie.
- Nie jest dopuszczalne zużycie kolorów na pojedyncze wierzchołki. W przypadku wystąpienia takiej sytuacji, albo nie uwzględniamy tego wierzchołku jeśli jest połączony ze wszystkimi innymi w grafie, albo zabieramy z innego turnusu (zawierającego co najmniej 3 wierzchołki) jakiś wierzchołek który nie jest z nim połączony. Razem będą tworzyli turnus spełniający wymagania.
- Ponieważ pracownicy którzy znają wszystkich swoich kolegów nie przyjmują udziału w turnusach (wierzchołki o stopniu V-1), to zostaną tacy pracownicy skreśleni z listy zanim rozpocznie się rozwiązywanie problemu.

Rozwiązanie problemu

Zostaną przedstawione dwa algorytmy rozwiązania problemu: "naiwne" podejście oraz algorytm heurystyczny LF (largest first). Niżej podano opis każdego algorytmu wraz z pseudokodem.

"Naiwne" podejście (aka Brute force)

Polega na sprawdzaniu każdego możliwego k-kolorowania grafu. K-kolorowanie to jest kolorowanie grafu używając k kolorów. Ze względu na to, że nie wiadomo jaka jest minimalna liczba kolorów potrzebnych do pokolorowania grafu algorytm musiałby sprawdzać kolorowania dla każdego k w zakresie od 1 do V (V – liczba wierzchołków). Jest to bardzo czasochłonne i dość trudne do wykazania poprawnej złożoności.
Ta standardowa wersja algortytmu została zmodyfikowana. Dla oszacowania liczby k jest wykorzystany algorytm Brona-Kerboscha do znajdowania największej maksymalnej kliki w grafie. Jest to skuteczne z

tego powodu, że istnienie kliki (podgrafu pełnego) o liczbie wierzchołków M narzuca minimalną liczbę potrzebnych kolorów M.

Pseudokod

\# **k** – liczba kolorów używanych do kolorowania
1. znajdź liczbę **k** korzystając z algorytmu **Brona-Kerboscha**
2. dla kazdego k-kolorowania grafu:
3. **jeśli** pokolorowanie jest dobre:
4. zwróć pokolorowanie

Złożoność obliczeniowa

- W przypadku zwykłego algorytmu Brute Force nie wiadomo ile wynosi k. Powoduje to, że sprawdzalibyśmy wszystkie pokolorowania dla k w zakresie od 1 do V. By tego nie robić liczba k $(k \leq V)$ jest wyznaczana za pomocą algorytmu Brona-Kerboscha, złożoność którego jest zależna od liczby wierzchołków w grafie i wynosi $\boldsymbol{O(3^{\frac{V}{3}})}$. Ta liczba wiąże się z maksymalnie możliwą liczbą klik w grafie.
- Liczba wszystkich k-kolorowań wynosi $\boldsymbol{O(k^V)}$. Wynika to z tego że dla każdego wierzchołku grafu mamy k sposobów na wybranie koloru.
- Sprawdzenie czy pokolorowanie jest dobre wymaga przejrzenia wszystkich krawędzi grafu, by się upewnić że żadna para sąsiednich wiechołków nie pokolorowana tym samym kolorem.
- Zatem złożoność k-kolorowania wynosi $\boldsymbol{O(k^V \cdot E)}$, gdzie E jest liczbą krawędzi w grafie.
- Ostatecznie złożoność algorytmu wynosi $\boldsymbol{O(3^{\frac{V}{3}} + k^V \cdot E)}$

Algorytm Largest First (aka Welsh-Powell)

Largest First (dalej WelshPowell) – algorytm heurystyczny przedstawiony przez panów Welsha i Powella, nie gwarantujący znalezienia optymalnego rozwiązania, natomiast dającego akceptowalny wynik w dość krótkim, w porównaniu do algorytmu Brute force, czasie. W tym algorytmie istotnym warunkiem dla dobrego kolorowanie jest kolejność kolorowanie wierzchołków. W odróżnieniu od standardowego algorytmu zachłannego, w którym pierwszy wierzchołek jest losowany, w algorytmie Welsh-Powell na początku dokonywano sortowanie listy wierzchołków grafu względem malejących stopni wierzchołków. Jest to istotne, gdyż od samego początku zmniejsza liczbę konfliktów podczas kolorowania, zapewniając mniejszą liczbę zużytych kolorów.

Pseudokod

1. posortuj wierzchołki wzlędem niemalejących stopni
2. pokoloruj pierwszy wierzchołek pierwszym kolorem
3. dla każdego wierzchołka **v** z pozostałych:
4. dla każdego sąsiada **v**:
5. **jeśli** sąsiad **v** już jest pokolorowany:
6. odznacz ten kolor jako taki którym nie możemy pokolorować **v**

7. pierwszym nieodznaczonym kolorem pokoloruj **v**

Złożoność obliczeniowa

Koszt sortowania **V** wierzchołków wynosi $\boldsymbol{O(V \cdot \log V)}$

Iterujemy w dwuch pętlach przech wszystkie wierzchołki więc złożoność wyniesie $\boldsymbol{O(V^2)}$

$\boldsymbol{O(V^2)}$ rośnie szybciej niż $\boldsymbol{O(V \cdot \log V)}$ a zatem ostateczna złożoność algorytmu wynosi $\boldsymbol{O(V^2)}$.