Dokumentacja wstępna

POP

Michał Brzeziński, Michał Sadlej

Zadanie: Sieć ograniczenia

Przy użyciu Algorytmu Ewolucyjnego zaprojektować sieć teleinformatyczną minimalizującą liczbę użytych systemów teletransmisyjnych o różnej modularności m, dla $(m = 1, m > 1, m >> 1)$.

Sieć opisana za pomocą grafu $G = (N,E)$, gdzie N jest zbiorem węzłów, a E jest zbiorem krawędzi.

Funkcja pojemności krawędzi opisana jest za pomocą wzoru $f_e(o) = \lceil o/m \rceil$.

Zbiór zapotrzebowań D, pomiędzy każdą parą węzłów opisuje macierz zapotrzebowań i jest dany.

Dla każdego zapotrzebowania istnieją co najmniej 2 predefiniowane ścieżki.

Sprawdzić jak wpływa na koszt rozwiązania agregacja zapotrzebowań, tzn. czy zapotrzebowanie realizowane jest na jednej ścieżce (agregacja), czy dowolnie na wszystkich ścieżkach w ramach zapotrzebowania (pełna dezagregacja).

Dobrać optymalne prawdopodobieństwo operatorów genetycznych oraz liczność populacji.

Dane pobrać ze strony http://sndlib.zib.de/home.action, dla sieci {\it polska}.

linki do danych:

https://sndlib.put.poznan.pl/download/sndlib-networks-native/polska.txt
https://sndlib.put.poznan.pl/home.action

link do opisu danych:

https://sndlib.put.poznan.pl/html/docu/io-formats/native/network.description.html

Wnioski

- Każde połączenie ma podstawową modułowość o danym koszcie z możliwością zainstalowania lepszej modułowości za odpowiednią dopłatą

- Rozwiązanie musi spełniać warunki z części DEMANDS

- ADMISSIBLE PATHS określa potencjalne ścieżki połączeń w DEMANDS

- Modularność - “pojemność” łącza

- Sieć opisana zbiorem węzłów i krawędzi

- Funkcja pojemności krawędzi - ceil(o/m)
- o - ilość danych do przesłania

- np
    - Jednostki w zadaniu nie zostały podane. Dla jasności przykładu użyto Mbps

    - potrzebne przesłanie o = 25 Mbps

    - modularność m = 10 Mbps

    - f_e(o) = ceil(25/10) = 3

    - potrzebna ilosć systemów transmicyjnych do przesłania wiadomości = 3

- Prawdopodobnie agregacja jest łatwiejsza do implementacji ale podatniejsza na przeciążenia przy skalowaniu w górę

Przykład

- Istnieją połączenia sieci Szczecin-Poznań-Wrocław i Bydgoszcz-Poznań-Wrocław
- Szczecin i Bydgoszcz muszą się połączyć z Wrocławiem i przesyłać 10 Mbps
- Mogą to robić wyłącznie przesyłając dane przez Poznań
- Jeśli modułowość krawędzi Poznań-Wrocław jest mniejsza niż 20 Mbps to nie będzie to możliwe

Założenia

- Modułowość można rozszerzyć, dopłacając.
- Nie można od razu wybudować rozszerzenia bez bazowej instalacji
- Rozwiązanie musi spełniać warunki z części DEMANDS

Realizacja

- Funkcja musi minimalizować koszta zapewniając dwie predefiniowane ścieżki

- Zostanie wykorzystany algorytm genetyczny albo ewolucyjny różniczkowy

- Generowane osobniki będą musiały spełniać warunki podane w DEMANDS i wymaganie zadania o dwóch predefiniowanych ścieżkach, musi to być brane pod uwagę zarówno w krzyżowaniu, mutacji jak i generacji osobników

- Należałoby stworzyć dwa generatory i funkcje krzyżujące, jeden priorytetujący agregację, a drugi dezagregację

- Osobnik będzie zapisywany w jeden z poniższych sposobów
    - Jako wektor o długości równej długości listy połączeń, którego elementem byłaby lista struktur mówiących o ilości przesyłanych danych i jakiego ADMISSIBLE_PATH jest częścią,

    - Albo wektor o długości równej długości listy wszystkich ADMISSIBLE_PATHS, struktur które przechowują ADMISSIBLE_PATH i ile jest nią przesyłane.

    - przykładowa struktura jednostki wektora dla drugiego przypadku:

      struct vector_cell_example:
          val ADMISSIBLE_PATH
          val sent_amount
          func whichDEMAND()
          func getUsedConnections()

- Generacja osobników przez losowe wybieranie ścieżek z możliwych z ADMISSIBLE_PATHS danego DEMAND i rozdzielenie losowo między nimi nakładu założonego przez dane DEMAND

- Funkcja kosztu zliczałaby sumę wydatków jakie trzeba by było zrobić żeby umożliwić podaną strukturę, oraz koszty przesłania danych DEMANDS.

- Wartość osobnika można by też zmniejszać, im więcej połączeń wykorzystuje w przypadku ze zwiększoną agregacją

- Mutacja wprowadzałaby zmiany w podziale pracy na ADMISSIBLE_PATHS danego DEMAND

- Krzyżowanie byłoby na przykład określeniem średniej arytmetycznej podziału na wszystkich ADMISSIBLE_PATHS danego DEMAND
    - np.
      są dwie ścieżki żeby wykonać demand “A-D 50”
      osobnik 1 wysyła 10 przez A-C-D i 40 przez A-B-D
      osobnik 2 wysyła 20 ścieżką A-C-D i 30 A-B-D
      ich potomek przesyłałby ((10+20)/50*2) przez A-C-D i 0.7*50 przez A-B-D

- Należałoby pewnie wprowadzić zaokrąglenie do liczb całkowitych albo nawet do dziesiątek, aby uniknąć problemu wielu małych przesyłek

- Inny pomysł na krzyżowanie mógłby brać rozwiązanie dla danego DEMAND które korzysta z mniejszej ilości ścieżek by umożliwić agregacje

- Należałoby wprowadzić ograniczenia by nie przeciążać danych połączeń. Może funkcja kosztu mogłaby zwracać wartość oznaczającą że rozwiązanie nie jest prawidłowe, żeby nie miało szansy się rozmnażać

Eksperymenty

- Sprawdzenie wydajności i wyników algorytmu którego generator i funkcja krzyżowania może generować dane faworyzujące agregację albo dezagregację w każdym układzie.
    - Na przykład jak radzi sobie algorytm z generatorem agregacyjnym i krzyżowaniem dezagregacyjnym w porównaniu z algorytmem z generatorem i krzyżowaniem agregacyjnym

- Eksperymenty związane z wykorzystywaniem rozbudowy. Na przykład wprowadzenie kary lub nagrody w funkcji kosztów za rozbudowywanie struktur ponad bazową wersję

- Porównanie algorytmu priorytetyzującego najkrótsze połączenia i algorytmu który na to nie uważa

- Porównanie algorytmu genetycznego i ewolucyjnego różniczkowego