Dokumentacja końcowa

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

- gen został zaimplementowany jako słownik o podanej strukturze

```
ADMISSABLE_PATHS = {                    # <- gen
    "Demand_0_1" : {                    # <- chromosom
        "Path_0": int(sent_amount)      # <- allel
        .
        .
        .
        'Path_6 ': int(sent_amount)
    },
    .
    .
    .
    "Demand_10_11" : {
        "Path_0": int(sent_amount)
        .
        .
        .
    },
}
```

- Stworzyliśmy klasę GeneticAlgorithm i DifferentialEvolutionAlgorithm posiadające -różne wersje metod takich jak mutacja czy krzyżowanie

Kod algorytmu genetycznego:

- generator
    - Tworzy populacje przy pomocy funkcji rand_split(number, num_of_parts, number_of_chunks). Funkcja ta rozdziela ilosć w demand (podaną jako number) na mniejsze fragmenty (number_of_chunks) i zwraca array o długości ilości możliwych ścieżek z losowo przydzielonymi elementami “chunks”

      Więcej fragmentów pomoże uzyskać równomierny rozkład dla każdego path, natomiast mniej większych fragmentów oznacza większą szansę na nierównówny podział obowiązków

- mutacje
    - Wybierane poprzez dodanie prawdopodobieństwa chcianej mutacji do inicjalizatora klasy (domyślnie wszystkie są równe 0)

    - mutate_for_aggregation - zabierający najmniejszą wartość przypisaną do path danego demand dodającą ją do innego path tego samego demand. Proces ten umożliwia skończenie z jednym z

    - mutate_without_aggregation - odejmuje od losowej wartośc jej część, opartą na _severity_of_mutation, z allelu i dodaje ją do innego allelu tego samego chromosomu

    - mutate_with_switch - zamieniający losowo dwie wartośći allelów jednego chromosomu

- krzyżowanie
    - wybierane przez podawaną przy inicjalizacji zmienną _cross_aggregating

    - cross_for_aggregation - wybiera którego rodzica chromosom weźmie na podstawie, tego które wykorzystują mniej zasobów (na podstawie ilości ścieżek i ilości linków w ścieżce)

    - cross_without_aggregation - allele są liczone na podstawie średniej obu rodziców

- rodzice są wybierani poprzez tournament selection

- funkcja kosztu evaluate_cost, zlicza
    - “setup cost” dla każdego linku
    - koszty rozbudowania o moduły potrzebne by rozwiązanie z genu działało
    - funkcje kary za każdy link gdzie rozwiązanie w genie przekroczyło pojemność największego modułu

Kod algorytmu różnicowego ewolucyjnego :

- generator
    - Korzysta z tego samego rodzaju generatora co algorytm genetyczny

- mutacja i krzyżowanie
    - Rodzice wybierani są losowo (z możliwością zmiany na wybór turniejowy)
    - Na podstawie współczynnika CR wybieramy czy chromosom zostanie wzięty od rodzica czy od mutacji, wykorzystując binarne podejście
    - w drugim przypadku chromosom staje się chromosomem rodzica zmienionym o wektor różnicy dwóch pozostałych rodziców dla każdego allelu w chromosomie

        - cały czas uważając by nie stracić części wyniku na zaokrągleniach, po mnożeniu wektora przez współczynnik F, oraz rozwiązując problem ujemnych wartości allelu metodą diffuse_negative_path()
        - po stworzeniu mutanta, porównywany jest z rodzicem i wybierane jest rozwiązanie z mniejszym wynikiem funkcji kosztu

- Dodatkowa mutacja:
    - By uniknąć problemu gdzie algorytm stabilizował się przed dojściem do akceptowalnego w założeniach wyniku, dodano dodatkową metodę mutacji smoothe_out, działającą podobnie do mutate_for_aggregation z algorytmu genetycznego, wprowadzając “czyściciela” chromosomów, przerzucając najmniejsze wartości do innego

Eksperymenty i Wyniki

- Opisy i wyniki eksperymentów zostały opisane w oddzielnym pliku: experiments.pdf