SK.POP.7 – Sieć, problem spełnialności ograniczeń

Maciej Wojno 302692
Jakub Zieliński 299041

1. Treść zadania

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) = ceil(o / m) 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 polska.

2. Pojęcia domenowe

2.1 Krawędź

Elementarna część grafu. Połączenie łączące dwa węzły w grafie. Każda krawędź ma swoją maksymalną pojemność.

2.2 Węzeł

Elementarna część grafu, w przypadku naszego problemu jest to miasto. Posiada informację o długości oraz szerokości geograficznej.

2.3 Modularność

Modularność określa maksymalną ilość zapotrzebowania, jaką można przepuścić przez krawędź, by zużyć jeden moduł (zwiększyć koszt rozwiązania o 1). Moduł zwiększa koszt niezależnie od stopnia wykorzystania, dlatego zadaniem optymalizacji jest zminimalizowanie ilości użytych modułów poprzez odpowiedni dobór odpowiednich ścieżek dla zapotrzebowań.

Przykład:

Modularność = 20
Przesyłane obciążenie = 22

Przez daną krawędź chcemy przepuścić obciążenie.
Zgodnie z formułą :

$$f_e(o) = \text{ceil}(o / m)$$

Koszt takiego rozwiązania to 2 moduły.

Jeżeli zmienilibyśmy wartość przesyłanego obciążenia do 20
np. przy pomocy dezagregacji to koszt wyniósłby 1 moduł (pod warunkiem, że ten sam problem nie zachodzi w na ścieżkach do których zostały oddelegowane pozostałe 2 jednostki obciążenia).

2.4 Zapotrzebowanie

Liczba jednostek obciążenia, która musi zostać przetransportowana między węzłem źródłowym a węzłem końcowym. Zapotrzebowanie jest realizowane przez predefiniowane ścieżki, jeżeli predefiniowana ścieżka realizuje część lub całość zapotrzebowania to każda krawędź wchodząca w skład ścieżki musi być w stanie “przyjąć” zadany przyczynek obciążenia. Relacja zapotrzebowania jest symetryczna.

2.5 Predefiniowana ścieżka

To zbiór krawędzi, które łączą węzeł źródłowy z węzłem końcowym. Nie posiada cykli.

3. Algorytm ewolucyjny

3.1 Model osobnika

Genom osobnika składa się z listy zapotrzebowań. Każde zapotrzebowanie to lista liczb reprezentujących sposób realizacji danego zapotrzebowania w zależności czy w zadaniu mamy agregację obciążeń, czy kompletną dezagregację.

Osobnik w zadaniu z agregacją

```
{
    "demands" : [
        {
            "demand_id" : "0-1",
            "paths_usage" : [0, 0, 0, 1, 0, 0, 0]
        },
        …
}
```

W powyższym przypadku postać osobnika oznacza, że zapotrzebowanie o id “0-1” zostanie zrealizowane przy użyciu wyłącznie predefiniowanej ścieżki o indeksie 3.

Osobnik w zadaniu bez agregacji

```
{
    "demands" : [
        {
            "demand_id" : "0-1",
            "paths_usage" : [0.2, 0.3, 0.1, 0.0, 0.4, 0.0, 0.0]
        },
        …
}
```

W tym przypadku zapotrzebowanie “0-1” zostanie zrealizowana zgodnie z proporcjami zawartymi w “paths_usage”. Dla przykładu ścieżki o indeksach 0 i 1 przetransportują odpowiednio 20% oraz 30% zapotrzebowania.

3.2 Parametry

3.2.1 Parametry zadania

- MODULARITY - wartość ilość obciążenia transportowana przez 1 moduł

- AGGREGATION - określa czy algorytm ma rozwiązać problem w którym zapotrzebowanie jest agregowane(1) lub rozproszone (0)

3.2.2 Parametry algorytmu

- POPULATION - liczba osobników w każdej kolejnej populacji

- CROSSOVER_PROB - prawdopodobieństwo krzyżowania

- MUTATION_PROB - prawdopodobieństwo mutacji osobnika

- MUTATION_POWER - siła mutacji jednego genu

- MUTATION_RANGE - ilość genów które zostaną zmutowane

- TARGET_FITNESS - docelowa wartość funkcji celu (warunek zatrzymania)

- MAX_EPOCHS - maksymalna liczba pokoleń (warunek zatrzymania)

- TOURNAMENT_SIZE - wielkość szranek selekcji turniejowej

- STALE_EPOCHS_LIMIT - ilość populacji bez poprawy najlepszego osobnika

3.3 Pseudokod

```
populations[0] = create_init_population(POPULATION)
t = 0
while continue(t, stale_generations_count, best_fitness):
  next_pop = []
  for i in range(POPULATION):
    if random.uniform(0, 1) < CROSSOVER_PROB:
      spec1 = select(populations[t])
      spec2 = select(populations[t])
      new_spec = mutation(crossover(spec1, spec2))
      next_pop += new_spec
    else:
      new_spec = mutation(select(populations[t]))
      next_pop += new_spec
  populations[t+1] = next_pop
  t += 1
```

3.4 Szczegóły procedur

3.4.1 Inicjalizacja

Procedura generująca początkowy zbiór osobników. W celu ustalenia rozmiaru bazowej populacji został użyty parameter POPULATION. Głównym zadaniem tej metody jest dobre optymalne “rozsianie” osobników po przestrzeni przeszukiwań.

3.4.2 Selekcja

Turniejowa, rozmiarem szranek będzie sterował parametr TOURNAMENT_SIZE. Prawdopodobieństwo selekcji danego osobnika zgodne z tym podanym na wykładzie.

3.4.3 Krzyżowanie

Z agregacją - po wylosowaniu rodziców, potomek powstaje poprzez wybranie dla każdego zapotrzebowania ścieżki losowo z jednego z rodziców.

Bez agregacji - nowy potomek powstaje przez uśrednienie odpowiednich wartości “paths_usage” z obu rodziców.

3.4.4 Mutacja

W danej iteracji, każdy osobnik ma prawdopodobieństwo mutacji równe MUTATION_PROB. Bez zwracania zostanie wybrane MUTATION_RANGE genów do mutacji. Jeżeli istnieje założenie o agregacji (AGGREGATION = 1) to mutacja odbywa się poprzez losową zmianę ścieżki wykorzystywanej do realizacji danego zapotrzebowania (gen = zapotrzebowanie). W przeciwnym przypadku dla danego genu jego mutacja postaci :

mut := normalize(mut + mut * mut_vect)

gdzie
mut_vect - wektor wartości z rozkładu U(0, MUTATION_POWER)

3.4.5 Zatrzymanie algorytmu

Zakończenie algorytmu następuje po spełnieniu co najmniej jednego z warunków ustawionych na początku działania programu :

- Liczba populacji z rzędu bez poprawy najlepszego osobnika przekroczy STALE_EPOCHS_LIMIT
- Uzyskanie osobnika uzyskującego wartość funkcji celu niższą lub równą niż TARGET_FITNESS
- Algorytm wykonał MAX_EPOCHS iteracji głównej funkcji

3.4.6 Funkcja celu

Ilość modułów potrzebna do zrealizowania wszystkich zapotrzebowań, wyliczana w następujący sposób.

Z agregacją:

```
moduły = 0
dla każdej krawędzi :
      obciążenie = 0
      dla każdego zapotrzebowania:
            jeśli wybrana ścieżka zawiera krawędź:
                  obciążenie += zapotrzebowanie
      moduły += ceil(obciążenie / modułowość)
zwróć moduły
```

Bez agregacji:

```
moduły = 0
dla każdej krawędzi :
      obciążenie = 0
      dla każdego zapotrzebowania:
            dla każdej ścieżki zapotrzebowania:
                  obciążenie += zapotrzebowanie * współczynnik ścieżki
      moduły += ceil(obciążenie / modułowość)
zwróć moduły
```

Będziemy dążyć do minimalizacji funkcji celu.

4. Badania

4.1 Schema

Tabela zawiera schemat badań. Każdy wiersz odpowiada testowanej konfiguracji algorytmu. W celu zminimalizowania wpływu losowości na znalezienie optimum globalnego algorytm będzie uruchamiany 10 razy dla każdej konfiguracji. Wynik każdego uruchomienia zostaną uśrednione.

Kombinacja parametrów Modularność i Agregacja, określa scenariusz w którym algorytm zostanie uruchomiony, z tego powodu te parametry zostały wyróżnione jako oddzielne kolumny.

Przetestujemy w jakim stopniu rozmiar populacji (np. 100, 500, 1000) wpływa na znajdowanie optimum zadania. Znajdziemy optymalne wartości prawdopodobieństw krzyżowania i mutacji. Sprawdzimy również jak założenie o agregacji wpływa na końcowy koszt rozwiązania. W tym celu porównamy wyniki dla różnych wartości Agregacji (tak / nie) przy ustalonych wartościach parametrów algorytmu [Pop, Cross, Mut, MutPwr] oraz ustalonej modularności.

Pop, Cross, Mut, MutPwr	Modularność	Agregacja	Optimum (średnia)	Odchylenie standardowe Optimum (średnia)	Czas wykonania (średnia)	Odchylenie standardowe czasu wykonania (średnia)
…	1	tak				
…	5	tak				

…	50	tak				
…	1	nie				
…	5	nie				
…	50	nie				

4.2 Rozszerzenie na inne modele sieci

W celu dodatkowego przetestowania zdolności algorytmu do znajdowania optymalnego rozwiązania, badania zostaną powtórzone na różnych modelach sieci (nie tylko na sieci polska). Inne modele nie nie posiadają określonych predefiniowanych ścieżek, więc zostaną one wyznaczone przez jeden z analitycznych algorytmów grafowych dostępnych w literaturze.

4.3 Porównanie z alternatywnym rozwiązaniem (dodatkowo)

Jeżeli po wykonaniu wcześniejszych punktów będziemy jeszcze dysponować wolnym czasem to planujemy wykonać porównanie zaimplementowanego przez nas algorytmu ewolucyjnego z alternatywnym rozwiązaniem (jeszcze nie wybranym). Porównanie będzie polegało na wykonaniu tej samej serii badań co w przypadku algorytmu ewolucyjnego, a następnie porównanie wyników.