[POP] - tworzenie planu egzaminów - dokumentacja wstępna

Piotr Gręda, Szymon Dyszewski

Grudzień 2022

1. Opis projektu

1.1. Krótki opis zadania:

Celem projektu jest opracowanie programu, który przy pomocy dokładnego wyszukiwania i algorytmu ewolucyjnego wygeneruje plan przeprowadzania egzaminów spełniającego poniższe założenia:

- Każdy egzamin trwa dokładnie 30 minut.
- Na każdym egzaminie musi znajdować się: Student, Przewodniczący, Opiekun, Członek komisji oraz Recenzent.
- Żaden z uczestników nie może znajdować się jednocześnie na więcej niż jednym egzaminie.
- Każdy egzamin musi być dopasowany do zadeklarowanej dostępności uczestników.
- Wszystkie zaplanowane egzaminy muszą się odbyć.
- Każdy z uczestników poza Przewodniczącym bierze udział w dokładnie jednym egzaminie.

1.2. Ograniczenia dostępności:

- Egzaminy odbywać się mogą od poniedziałku do piątku od godziny 10 do godziny 13.
- Studenci dostępni są cały tydzień z odstępstwem rzędu 4 - 6 slotów czasowych (slot czasowy - 30 minut).
- Przewodniczący dostępny jest przez wszystkie sloty czasowe jednego dnia.
- Opiekun, Członek komisji oraz Recenzent każdego dnia mają do dyspozycji 2 sloty czasowe.

1.3. Założenia dotyczące ilości osób zaangażowanych:

- Studenci - ok. 30
- Przewodniczący - ok. 5
- Opiekuni - ok. 30
- Członkowie komisji - ok. 30
- Recenzenci - ok. 30

2. Wykorzystane algorytmy

2.1. Opis

Wymagane jest użycie metody dokładnego przeszukiwania oraz algorytmu ewolucyjnego.

- Kryterium sukcesu - spełnienie wszystkich założeń, które zostały dane nam przez prowadzącego i które zapisane są powyżej.
- Metoda oceny jakości rozwiązania - Jest to liczba egzaminów, których nie udało się przeprowadzić. Taki sposób jej skonstruowania umożliwia poruszanie się wzdłuż gradientu tej funkcji.

- Aby zwiększyć szanse na "szybkie" znalezienie ekstremum spełniającego wymagania zastosujemy w naszym algorytmie ewolucyjnym selekcję turniejową, skutecznie stawiając na eksplorację.
- W naszym przypadku populacją jest plan egzaminów, chromosomem osobnika jest zestawienie wszystkich uczestników egzaminu, a genem jest pojedyncza osoba.
- Algorytm ewolucyjny będzie generował nam kolejne pokolenia populacji, krzyżując geny w pojedynczym chromosomie (mieszając uczestników egzaminu). Mutacja jest niewskazana, nie możemy pozwalać niektórym pracownikom odpoczywać, ponieważ inni musieliby mieć dwóch studentów, a to zakazane.
- Aby algorytm był kompletnym, to w każdym kroku tworzenia ścieżki będzie on sprawdzał, czy nie została utworzona już ścieżka, która miałaby mniejszą funkcję kosztów (opierając się na heurystyce).

2.2. Przykładowe obliczenia

- Selekcja turniejowa:

$$p_s(P_f^t) = \frac{1}{\mu^s}((\mu - i + 1)^s - (\mu - i)^s)$$

  Przebieg:

  - Populacja bazowa: osobniki o ocenie 3, 5, 7
  - Szranki na 2 osobniki
  - Losuję do szranek: osobniki 5 i 7, wygrywa 7 → osobniki 3 i 3, wygrywa 3 → osobniki 7 i 7, wygrywa 7
  - Nasza populacja jest zastępowana przez [7, 3, 7].
  - Zastosowanie takiego algorytmu sprzyja eksploracji, istnieje szansa, że po sukcesji do następnej generacji przejdzie punkt, który uchodzi za słabszy (w tym przypadku 3). Taki punkt, daje nam szansę na "odkrycie" nowego, nieoczywistego optimum.

- Krzyżowanie:

5	10	3	12	30	+	24	11	9	21	25	=	24	11	3	12	30	+	5	10	9	21	25

  Przebieg:

  - Początkowo mamy dwa chromosomy (egzaminy) o określonych powyżej genach (uczestnikach).
  - Następnie tworzymy z nich nowe chromosomy zamieniając poszczególne geny pierwotnych chromosomów.

3. Plan eksperymentów

3.1. Poprawność rozwiązań

Ważnym elementem będzie upewnienie się, że algorytm zawsze znajduje rozwiązanie, jeśli takie istnieje. W tym celu dla odpowiednio dopasowanych danych będziemy weryfikować nie jakość rozwiązań, a ich słuszność.

3.2. Funkcja heurystyczna

Do przeprowadzenia eksperymentów wykorzystamy skrypty do generowania danych pseudolosowych o różnych rozmiarach. Dla poszczególnych danych zostaną wykonane serie testów mające na celu zbliżenie nas do ustalenia jak najlepszej funkcji heurystycznej.

3.3. Złożoność algorytmu

Ostatnim celem eksperymentów będzie optymalizacja krzyżowania czy selekcji, by algorytm nie tracił czasu, błądząc w miejscu.