Przeszukiwanie i optymalizacja

Dokumentacja wstępna

Dominika Boguszewska
Aleksander Szymczyk

Semestr 24Z

Spis treści

1 Temat projektu - Santa's Workshop Tour 2019

Kolejny z problemów, w którym Mikołaj potrzebuje pomocy, tym razem przy planowaniu wizyt w swoim warsztacie. Jako że liczba chętnych jest ogromna, a Mikołaj jest miły, zdecydował się zaprosić 5000 rodzin i pozwolić im na samodzielne wybranie preferencji - kiedy chcieliby go odwiedzić. Po otrzymaniu zgłoszeń, Mikołaj uświadomił sobie, że nie jest w stanie zapewnić wszystkim ich preferowanych terminów. Zdecydował się więc na wypłacanie rekompensaty w zależności od tego, jak bardzo pasuje danej rodzinie przydzielony termin. Pomóż Mikołajowi zminimalizować koszt rekompensaty za niedogodne terminy wizyty.

Dane: 5000 rodzin

Dokładne informacje i dane: https://www.kaggle.com/c/santa-workshop-tour-2019

2 Analiza danych

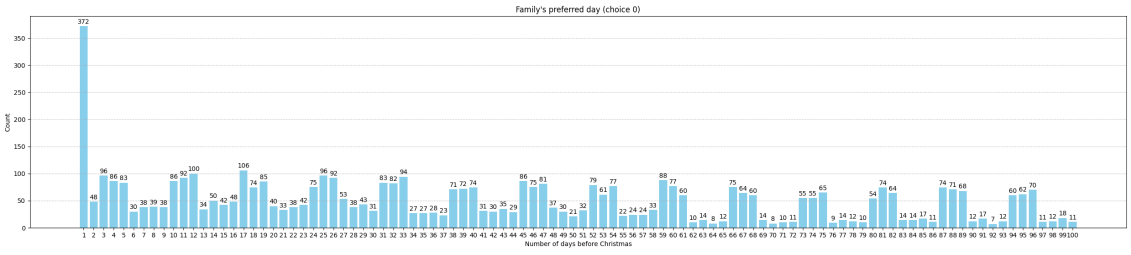
Zbiór danych zawarty w pliku *family_data.csv* opisuje preferencje 5000 rodzin, zidentyfikowanych za pomocą indeksów od 0 do 4999. Dla każdej rodziny podano ich preferowane daty odwiedzin, zapisane w kolumnach `choice_0`, `choice_1`, ..., `choice_9`. Wartości w tych kolumnach to liczby całkowite z zakresu od 1 do 100, oznaczające liczbę dni przed Bożym Narodzeniem. Na przykład wartość 1 odpowiada dacie 24 grudnia, wartość 2 to 23 grudnia, itd.

Preferencje te definiują 10 najbardziej pożądanych dat, kiedy każda rodzina chciałaby odwiedzić Warsztat Świętego Mikołaja. Dodatkowo, dla każdej rodziny podano wartość `n_people`, określającą liczbę osób, które planują odwiedzić Warsztat.

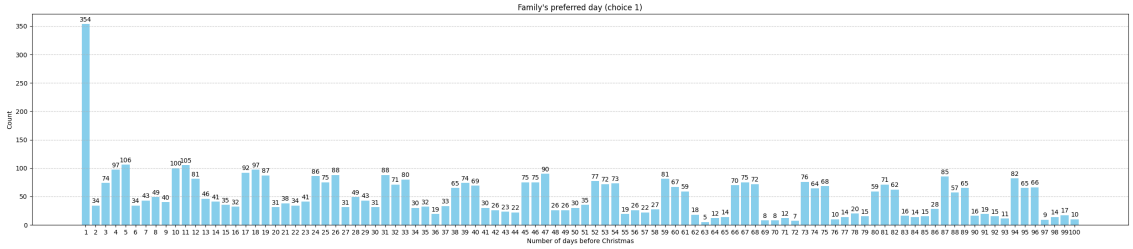
W celu analizy danych, dla każdej ze 100 dat zostanie policzona liczba rodzin, które wybrały daną datę jako swój pierwszy wybór, liczba rodzin, które wybrały ją jako drugi wybór itd. Poza liczbą rodzin policzony zostanie również rozkład liczności rodzin. Zamierzamy przeanalizować także wzrost kosztu w zależności od przyjętego wyboru.

2.1 Dni preferowane przez rodziny

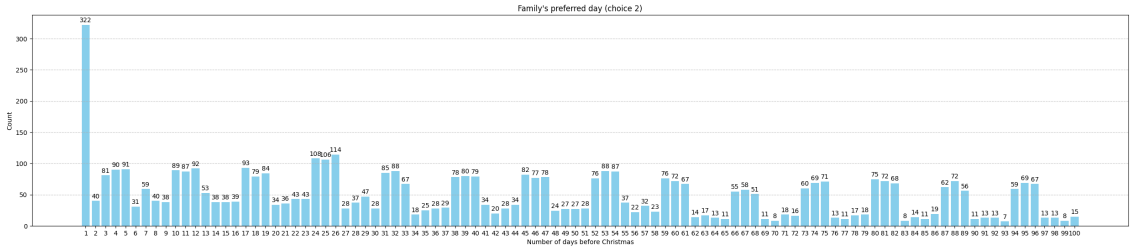
Poniżej przedstawiono histogramy obrazujące rozkład rodzin wybierających poszczególne dni według ich preferencji `choice_0`, `choice_1` itd.



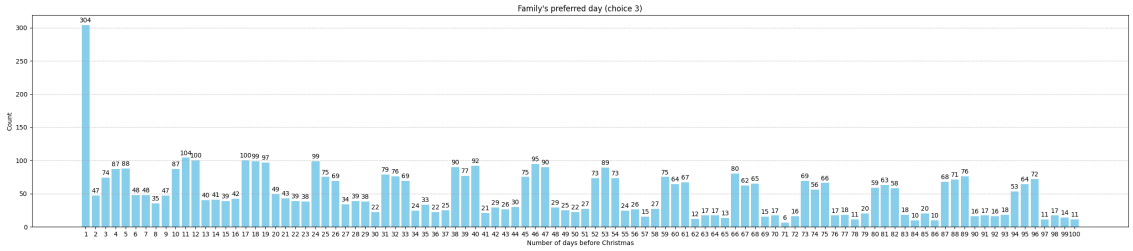
Rysunek 1: Preferowane dni - wybór 0



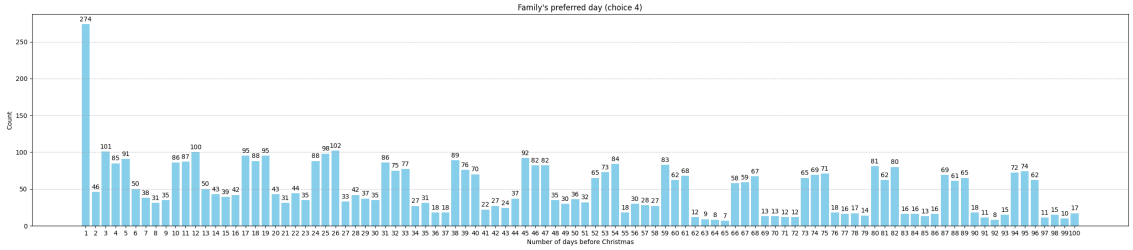
Rysunek 2: Preferowane dni - wybór 1



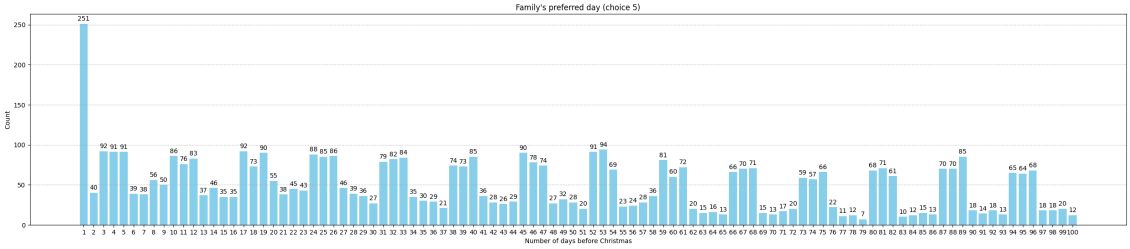
Rysunek 3: Preferowane dni - wybór 2



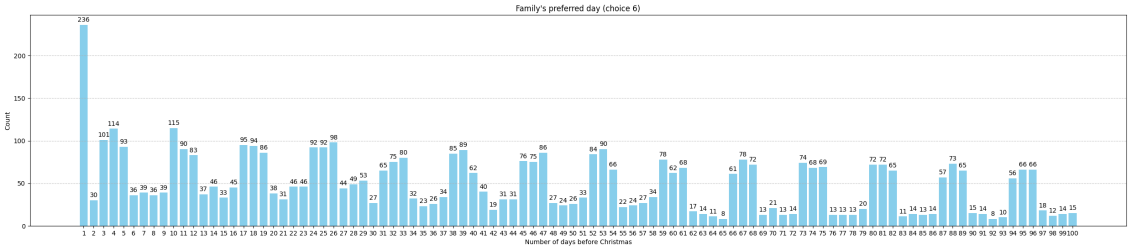
Rysunek 4: Preferowane dni - wybór 3



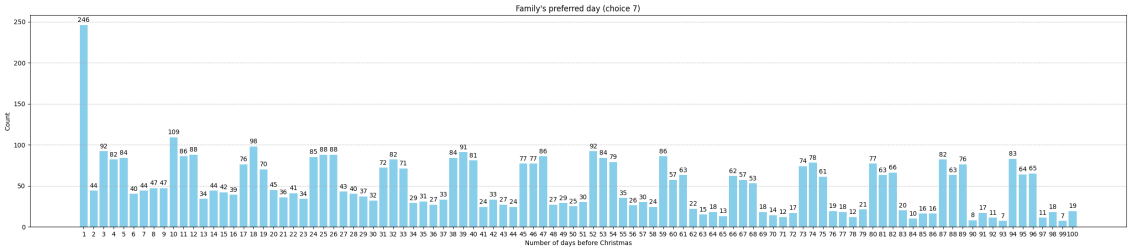
Rysunek 5: Preferowane dni - wybór 4



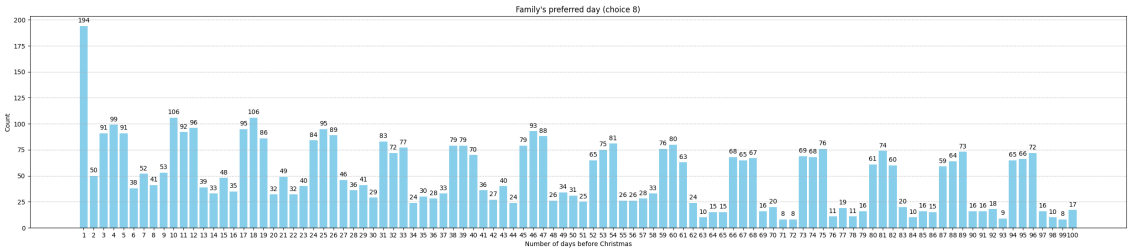
Rysunek 6: Preferowane dni - wybór 5



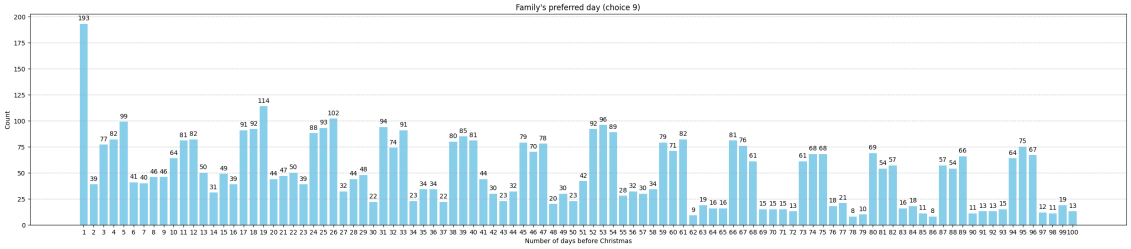
Rysunek 7: Preferowane dni - wybór 6



Rysunek 8: Preferowane dni - wybór 7



Rysunek 9: Preferowane dni - wybór 8

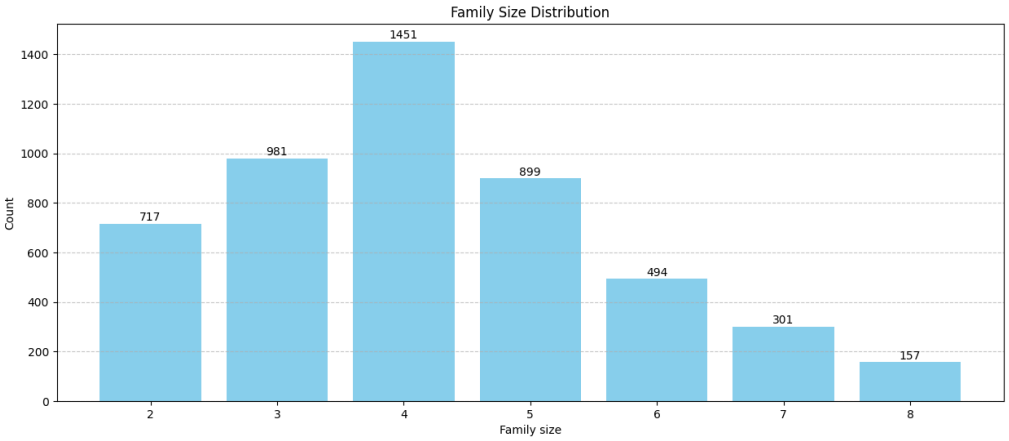


Rysunek 10: Preferowane dni - wybór 9

Jak możemy zauważyć, najchętniej wybierany był 24 grudnia, czyli 1 dzień przed Świętami. W drugiej kolejności rodziny wybierały piątki, soboty bądź niedziele (zakładając, że posługujemy się kalendarzem z 2019 roku).

2.2 Rozkład liczebności rodzin

Poniżej przedstawiono histogram z rozkładem liczby członków rodzin.

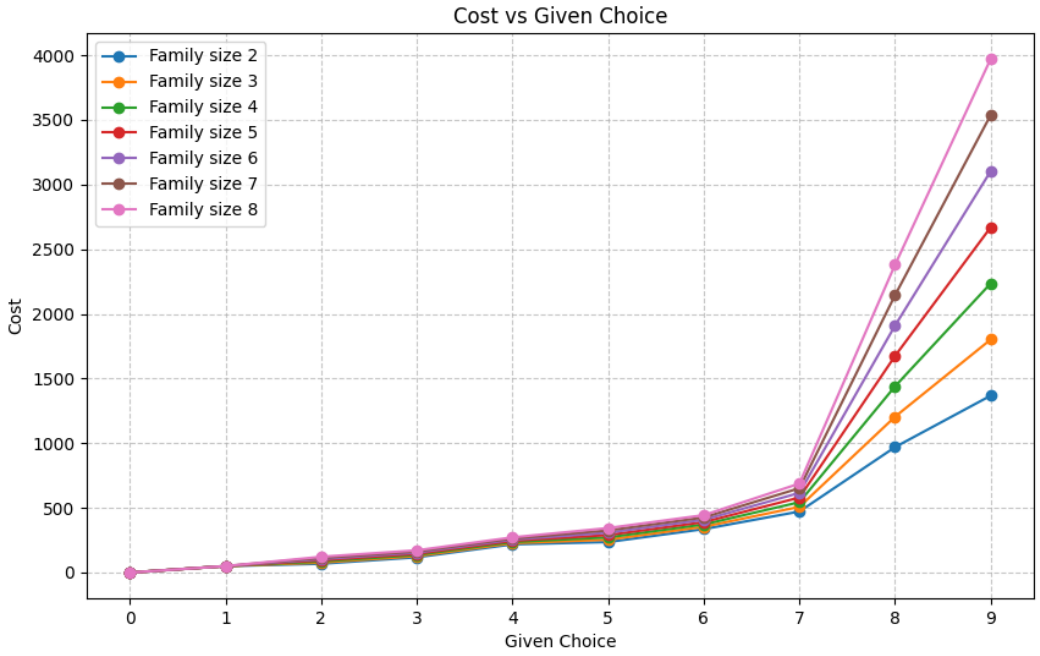


Rysunek 11: Rozkład liczebności rodzin

Jak możemy zauważyć, najwięcej jest rodzin 4-osobowych, a najmniej jest rodzin z dużą liczbą członków.

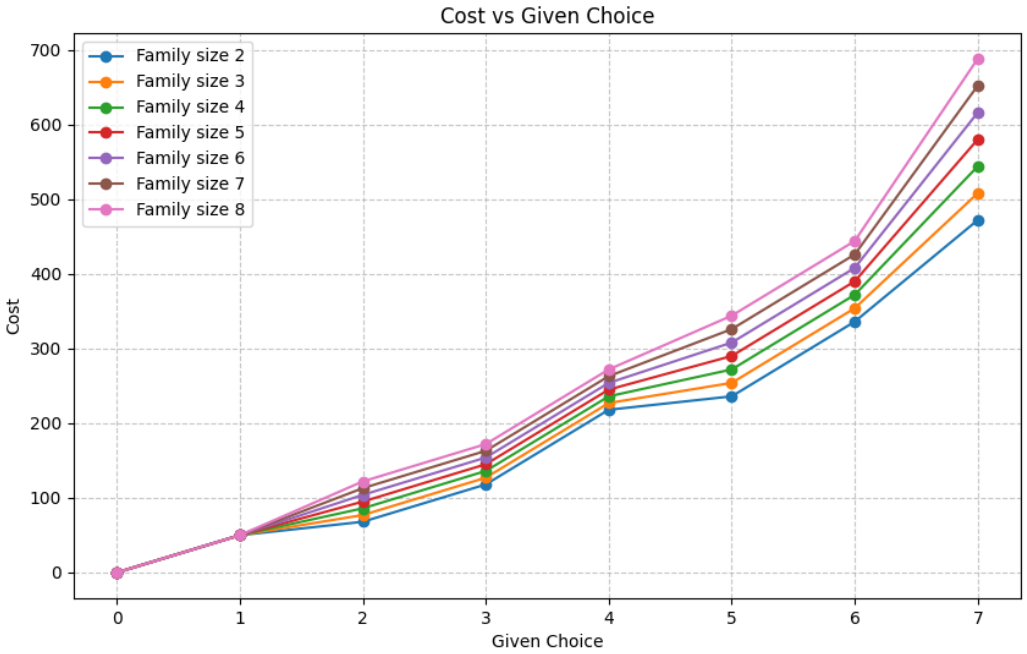
2.3 Wzrost kosztu w zależności od przyjętego wyboru i liczebności rodzin

Poniżej został pokazany wykres porównujący naliczone kary dla każdej z liczebności rodzin oraz dla każdego możliwego przypisanego im wyboru.



Rysunek 12: Wzrost kosztu w zależności od przyjętego wyboru i liczebności rodzin

Jak możemy odczytać z wykresu, od wyboru 7 naliczana kara zaczyna drastycznie rosnąć, zatem powinniśmy unikać przypisywania rodzinom wyborów powyżej 7.



Rysunek 13: Kara za odwiedzającą rodzinę do wyboru 7

Jak możemy odczytać z wykresów, najlepiej byłoby przypisywać rodzinom wybory 0-3. Możemy także zauważyć, że dla małych rodzin przyrost wartości kary od wyboru 4 do 5 jest niewielki.

3 Wstępna propozycja rozwiązania

3.1 Algorytm ewolucyjny

3.1.1 Reprezentacja osobnika i populacja

Każdy osobnik jest reprezentowany jako tablica o wymiarach `5000 × 1`, gdzie każda komórka określa dzień (z zakresu od 1 do 100), w którym dana rodzina odwiedzi Warsztat.

Dla każdego osobnika tworzona jest dodatkowo mapa dni, w której przechowywana jest aktualna liczba odwiedzających dla każdego dnia (początkowo ustawiona na 0).

Dla każdej rodziny przypisywane są dni odwiedzin zgodnie z poniższymi zasadami:

1. **Przypisanie dnia:**
   W celu ograniczenia tworzenia populacji, w których znaczna liczba osobników narusza narzucone restrykcje, inicjalizacja nie będzie całkowicie losowa.

   - Iterujemy po wszystkich rodzinach osobnika.
   - Dla każdej rodziny próbujemy przypisać kolejny preferowany dzień, zaczynając od najbardziej preferowanego.

- Sprawdzamy ograniczenia liczby osób przypisanych do każdego dnia:
  - Jeżeli wybrany dzień nie łamie ograniczeń (czyli liczba osób po przypisaniu rodziny nie przekroczy 300 osób), rodzina przypisywana jest do tego dnia.
  - Jeżeli wybór łamałby ograniczenia, przechodzimy do kolejnej preferencji rodziny.
  - Jeśli żadna z preferencji nie może zostać użyta, wybierany jest dzień z najmniejszą liczbą odwiedzających.

2. **Obsługa dni z niedoborem odwiedzających (< 125 osób):**
   - Dla każdego dnia, który nie spełnia minimalnego limitu odwiedzających, wykonywane są następujące kroki:
     - Przeszukiwana jest lista preferencji rodzin, które zostały przypisane do jednego z 10 dni o największej liczbie odwiedzających.
     - Jeśli dzień z niedoborem znajduje się w preferencjach którejkolwiek rodziny, jej przypisanie zostaje zmienione na ten dzień.
     - Jeżeli nie jest to możliwe, losowa rodzina z najbardziej obciążonego dnia zostaje przeniesiona do dnia z niedoborem.
   - Proces ten powtarzany jest, aż wszystkie dni spełnią ograniczenia.
3. **Losowa kolejność:**
   - Kolejność rozpatrywania rodzin jest losowa dla każdego osobnika, aby zwiększyć różnorodność populacji.
4. **Tworzenie populacji:**
   - Proces przypisywania dni powtarzany jest dla każdego z `N` osobników, gdzie `N` to hiperparametr określający wielkość populacji.

3.1.2 Krzyżowanie

W algorytmie zastosowano krzyżowanie jednopunktowe, w którym rodzice są wybierani za pomocą selekcji turniejowej.

1. **Wybór rodziców do krzyżowania przy pomocy selekcji turniejowej**
   - Losowo wybierana jest grupa k osobników z populacji.
   - Następnie spośród nich, na podstawie wartości funkcji kosztu, wybierany jest najlepszy osobnik, który zostanie rodzicem.
   - Proces ten powtarza się, aż wybrana zostanie odpowiednia liczba rodziców (hiperparametr `parents`).
2. **Przebieg krzyżowania**
   - W zadaniu zastosowano **krzyżowanie jednopunktowe**.
   - Potomek dziedziczy część harmonogramu przed punktem podziału od jednego rodzica, a pozostałą część od drugiego rodzica.

3.1.3 Mutacja

Proces mutacji w opisywanym zadaniu został zaprojektowany w dwóch wariantach, które są stosowane w różnych fazach działania algorytmu:

- **Wariant eksploracyjny (początkowa faza algorytmu):**
  Na wczesnym etapie działania algorytmu dla losowo wybranych osobników zmienia się przypisany rodzinie dzień na dowolną wartość z zakresu 1–100. Celem tego wariantu jest zwiększenie eksploracji przestrzeni rozwiązań

- **Wariant eksploatacyjny (końcowa faza algorytmu):**
  W późniejszych iteracjach algorytmu wariant eksploracyjny zostaje zastąpiony przez wariant eksploatacyjny. W tym wariancie dla losowo wybranych osobników przypisany rodzinie dzień zostaje zmieniony na jeden z 10 preferowanych przez nią dni.

Proces mutacji można podzielić na następujące kroki:

1. **Wybór osobników do mutacji**
   - Każdy osobnik w populacji ma prawdopodobieństwo P_m, zostania wybranym do mutacji.
2. **Przebieg mutacji (Wariant eksploracyjny)**
   - Dla wybranego osobnika, każda z rodzin ma niewielkie prawdopodobieństwo P_{mf} na zmianę przypisanego dnia na inny.
   - Nowy dzień jest losowany z zakresu 1-100.
3. **Przebieg mutacji (Wariant eksploatacyjny)**
   - Dla wybranego osobnika, każda z rodzin ma niewielkie prawdopodobieństwo P_{mf} na zmianę przypisanego dnia na inny.
   - Nowy dzień jest losowany spośród dni preferowanych przez tę rodzinę, co pozwala zwiększyć dostosowanie osobnika do preferencji rodzin.

3.1.4 Sukcesja

W celu zapewnienia utrzymania najlepszych osobników wśród tych przechodzących do następnej generacji zastosowana zostanie selekcja elitarna.

1. **Obliczenie wartości funkcji kosztu dla osobników**
   - Poza inicjalizacją populacji startowej algorytm nie koryguje wartości osobników, tak by spełniały ograniczenia. Jest to spowodowane wysokim kosztem obliczeń związanym z takim procesem przy każdej iteracji.
   - W ramach alternatywy funkcja kosztu przyjmuje skrajnie duże wartości dla osobników łamiących ograniczenia, efektywnie powodując eliminacje takich osobników w ramach postępu algorytmu.
2. **Selekcja elitarna**
   - Osobniki są sortowane według wartości funkcji celu.
   - `S` najlepszych osobników kopiowanych jest bez zmian do następnej populacji.

- Pozostała część populacji jest uzupełniana przez nowe osobniki powstałe w wyniku krzyżowania i mutacji.

3.2 Weryfikacja

W celu weryfikacji poprawności algorytmu, wstępne testy zostaną przeprowadzone na ograniczonym zbiorze danych, składającym się ze 100 rodzin. Po osiągnięciu na nim wystarczająco niskiej wartości funkcji celu (oceniona w porównaniu do wyników zgłoszeń na platformie Kaggle) eksperymenty zostaną kontynuowane na pełnym zbiorze danych.

Jeżeli po przeprowadzonych eksperymentach algorytm nadal będzie miał trudności z generowaniem osobników spełniających narzucone ograniczenia, do poszczególnych etapów rozwiązania (krzyżowania, mutacji, selekcji) stopniowo będą dodawane mechanizmy wymuszające ich spełnienie.

4 Definicja funkcji kosztu

- `restriction_penalty`: Każdego dnia zwiedzania w Warsztacie musi znajdować się 125-300 osób. Za każdy dzień niespełniający tych ograniczeń do funkcji kosztu dodawana jest kara o wartości 100000. Jeżeli wszystkie dni spełniają ograniczenia kara wynosi 0.
- `choice_penalty`: W zależności od zaakceptowanego wyboru naliczane są poniższe kary:
  - `choice_0`: brak kary
  - `choice_1`: 50
  - `choice_2`: 50 + 9 * rozmiar_rodziny
  - `choice_3`: 100 + 9 * rozmiar_rodziny
  - `choice_4`: 200 + 9 * rozmiar_rodziny
  - `choice_5`: 200 + 18 * rozmiar_rodziny
  - `choice_6`: 300 + 18 * rozmiar_rodziny
  - `choice_7`: 300 + 36 * rozmiar_rodziny
  - `choice_8`: 400 + 36 * rozmiar_rodziny
  - `choice_9`: 500 + 235 * rozmiar_rodziny
  - `otherwise`: 500 + 434 * rozmiar_rodziny
- `accounting_penalty`: Kara księgowa:

$$\sum_{d=100}^{1} \frac{(N_d - 125)}{400} N_d^{(\frac{1}{2} + \frac{|N_d - N_{d+1}|}{50})}$$

  gdzie N_d to liczba osób odwiedzająca Warsztat danego dnia.
- W pierwszy dzień (czyli 100, ponieważ liczymy od tyłu do świąt), $N_{100} = N_{101}$
- Całkowita wartość funkcji kosztu to: `restriction_penalty + choice_penalty + accounting_penalty`

5 Sposób mierzenia jakości rozwiązania

Program generuje rozwiązania w formacie zgodnym z przykładowym rozwiązaniem, które znajduje się w pliku *sample_submission.csv* dostępnym na stronie zadania. Jest to plik tekstowy, w którym w pierwszej kolumnie znajduje się indeks rodziny, a w drugiej dzień, w który ta rodzina odwiedzi Warsztat Świętego Mikołaja.

Aby ocenić jakość uzyskanego rozwiązania, najpierw weryfikujemy jego poprawność, sprawdzając, czy liczba osób odwiedzających Warsztat każdego dnia zawiera się w przedziale od 125 do 300. Dodatkowo mierzymy czas potrzebny na znalezienie rozwiązania, aby ocenić, czy algorytm jest wystarczająco efektywny i działa w akceptowalnym czasie.

Podczas porównywania różnych rozwiązań, wybieramy to, które ma niższą wartość funkcji kosztu. Oprócz wartości funkcji kosztu, porównujemy także czasy potrzebne na znalezienie poszczególnych rozwiązań, aby ocenić, czy poprawa jakości rozwiązania jest opłacalna w kontekście czasu potrzebnego na jego znalezienie.