# Podstawy Sztucznej Inteligencji - przeszukiwanie

Marcin Białek

## Wstęp

Niniejszy dokument przedstawia analizę wyników algorytmu genetycznego, który stara się rozwiązać następujące zadanie:

> *Opracować algorytm genetyczny służący do optymalnego rozmieszczenia N weselników przy M stołach weselnych każdy z N/M miejscami, gdzie N/M jest liczbą całkowitą. Każdy z weselników lubi każdego innego weselnika w innym stopniu. Staramy się tak rozmieścić gości, aby łączne zadowolenie mierzone sumą sympatii pomiędzy weselnikami było jak największe.*

Analizowany jest wpływ parametrów (tj. rozmiar populacji, prawdopodobieństwo mutacji) oraz danych (tj. liczba osób, liczba stołów) na jakość rozwiązania oraz czas, który był potrzebny na jego znalezienie. Czas oznacza tutaj liczbę generacji. Zakłada się, że wzrost rozmiaru populacji poprawi jakość rozwiązania i przyspieszy jego znalezienie, zwiększenie prawdopodobieństwa mutacji wpłynie korzystnie na właściwości eksploracyjne algorytmu (co przełoży się na lepsze rozwiązanie), natomiast zwiększenie liczby osób wydłuży czas poszukiwania, a zwiększenie liczby stołów poprawi jakość rozwiązania.

## Słownik

W dalszej części tekstu stosowane są następujące terminy i zmienne:

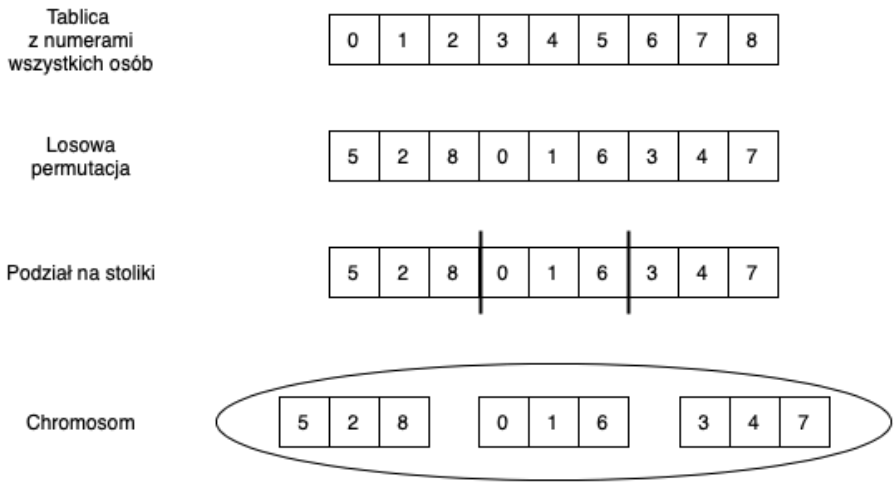
- chromosom - możliwe rozwiązanie problemu, konfiguracja osób przy stole,
- generacja - iteracja algorytmu genetycznego,
- poziom sympatii - liczba z przedziału [0, 1] mówiąca jak bardzo osoba A lubi osobę B, przy czym 0 oznacza „w ogóle nie lubi” a 1 „bardzo lubi”,
- poziom zadowolenia - suma poziomów sympatii osób siedzących przy jednym stole,
- wartość funkcji dopasowania/ocena - liczba z przedziału [0, 1] mówiąca jak dobre rozwiązanie problemu reprezentuje dany chromosom, im mniejsza wartość, tym lepiej,
- N - liczba wszystkich osób/weselników,
- M - liczba wszystkich stołów,
- L - macierz kwadratowa o wymiarze N zawierająca poziomy sympatii. Wartość w wierszu a i kolumnie b odpowiada poziomowi sympatii osoby numer a do osoby numer b. Wartości na przekątnej (czyli poziom sympatii do samego siebie) nie są wykorzystywane i mogą być dowolne,
- $Z_{max}$ - teoretycznie maksymalny poziom zadowolenia, tzn. poziom zadowolenia dla stolika, przy którym poziom sympatii każdej z osób do każdej innej jest równy 1,
- $\mu$ - rozmiar populacji, liczba chromosomów w populacji,
- $\sigma$ - prawdopodobieństwo mutacji.

## Algorytm

Do rozwiązania problemu został zaprojektowany algorytm genetyczny, w którego skład wchodzą: inicjalizacja, ocena (ewaluacja), selekcja, krzyżowanie, mutacja, sukcesja. Każdy z etapów jest opisany poniżej. Algorytm przyjmuje dane (N, M, L) oraz ustawienia (μ, σ), a kończy działanie kiedy średnia wartość funkcji dopasowania nie zmieni wartości przez 100 generacji (zakłada się wtedy, że lepsze rozwiązanie już nie zostanie znalezione).

### Inicjalizacja

Każdy z chromosomów tworzony jest poprzez stworzenie listy wszystkich numerów osób, czyli liczb od 0 do N-1. Lista jest losowo permutowana, a następnie dzielona na M części, każda o rozmiarze N/M. Jedna część reprezentuje jeden stolik, a jej zawartość to osoby siedzące przy nim. Proces powtarzany jest μ razy generując w ten sposób początkową populację (generację 0).




Obliczany jest również maksymalny poziom zadowolenia zgodnie ze wzorem:

$$Z_{max} = \frac{N}{M}(\frac{N}{M} - 1)$$

### Ocena (ewaluacja)

Dla każdego chromosomu obliczany jest poziom zadowolenia każdego ze stołów. Następnie wybierany jest najmniejszy poziom zadowolenia ($z_{min}$) i z jego pomocą obliczana jest wartość funkcji dopasowania (ocena chromosomu) według wzoru:

$$q = 1 - \frac{z_{min}}{Z_{max}}$$

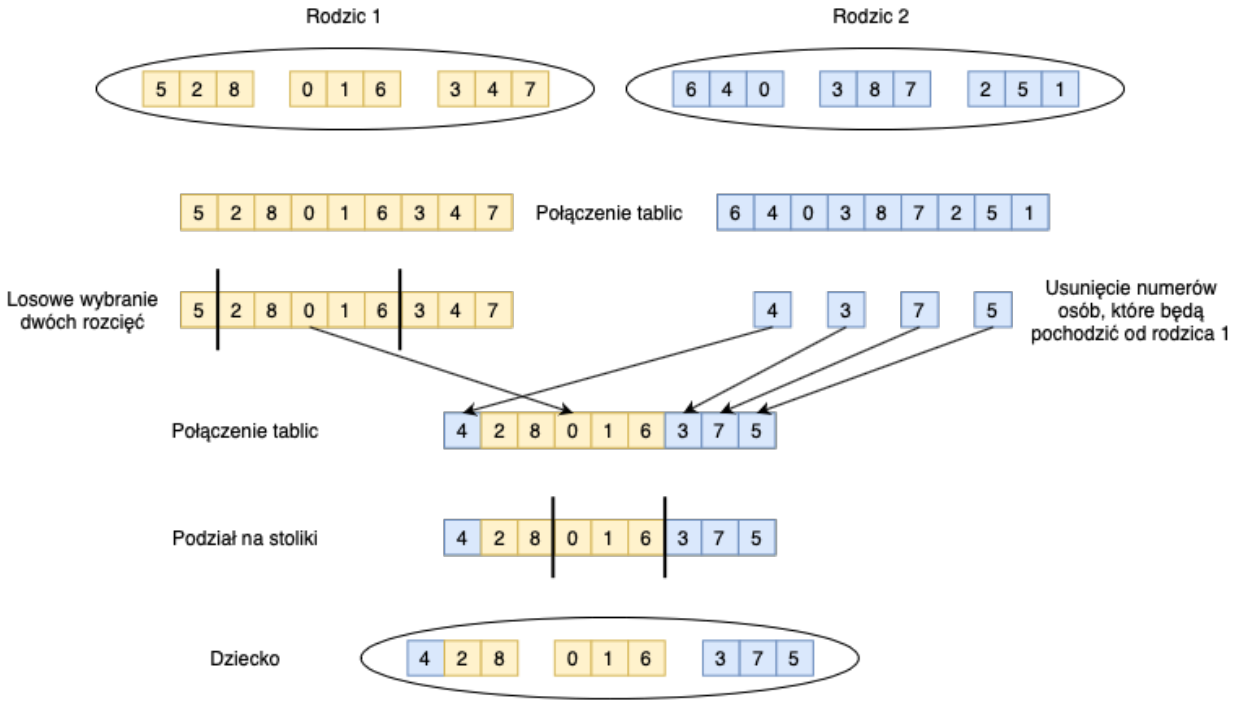
Wybieranie minimalnego poziomu zadowolenia zamiast np. sumy pozwala uniknąć sytuacji, w której poziom zadowolenia jednego ze stolików jest bliski 0, a poziom zadowolenia innego stolika jest bardzo wysoki, więc ogólna ocena chromosomu jest dość dobra. Innymi słowy pozwoli to uniknąć zbyt dużej różnicy poziomów zadowolenia w danym chromosomie.

### Selekcja

Generowane jest μ losowych par chromosomów-rodziców. Prawdopodobieństwo wyboru danego chromosomu jest wprost proporcjonalne do jego oceny: im lepsza wartość funkcji dopasowania, tym większe prawdopodobieństwo (reprodukcja ruletkowa).

### Krzyżowanie

Dla każdej pary wybranej podczas selekcji przeprowadzane jest zmodyfikowane krzyżowanie dwupunktowe. Tablice przechowujące numery osób przy stołach są łączone. Następnie losowo wybierane są dwa punkty rozcięcia dla pierwszego rodzica, co dzieli tablicę na trzy części. Środkowa część w całości przechodzi do dziecka, a brakujące numery osób są uzupełniane po kolei z tablicy drugiego rodzica.




### Mutacja

Dla każdego stworzonego dziecka z prawdopodobieństwem σ zamienia się dwa losowo wybrane numery osób.

### Sukcesja

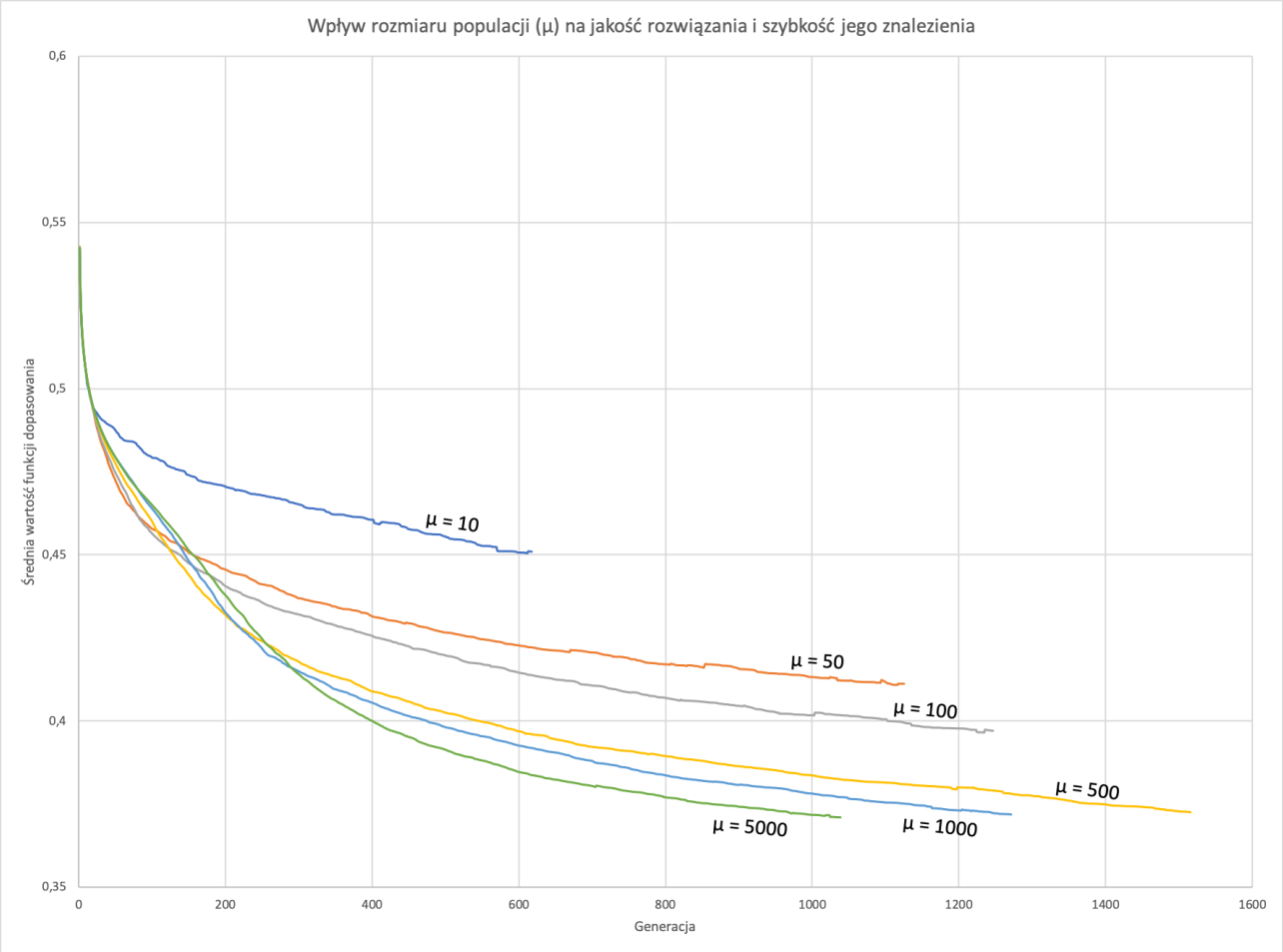
Populacja obecnej generacji łączona jest ze stworzonymi na etapie krzyżowania dziećmi. Chromosomy są sortowane ze względu na ocenę, a spośród nich wybierane jest μ najlepszych (sukcesja elitarna). Wybrane osobniki tworzą populację następnej generacji.

## Implementacja

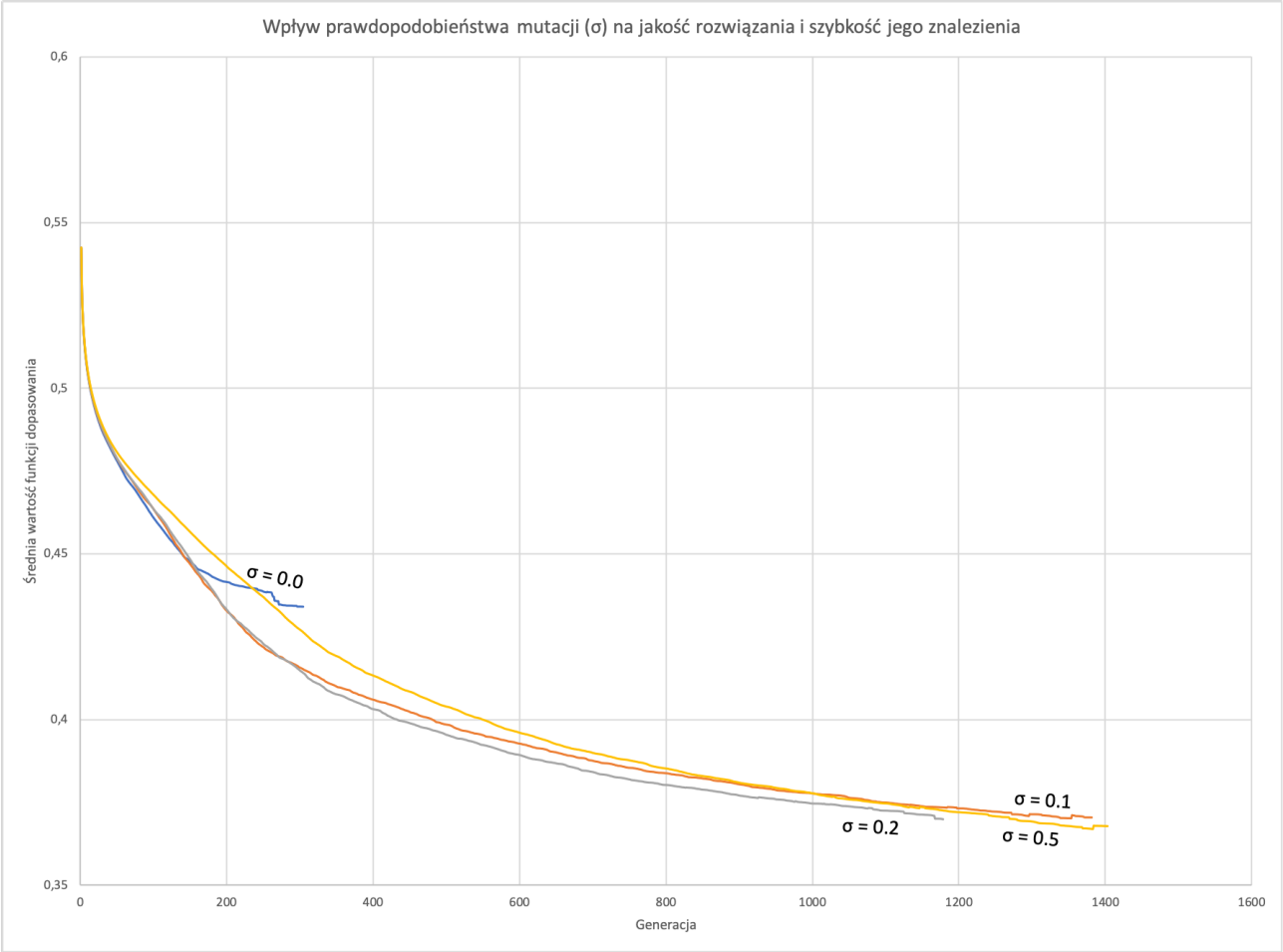
Algorytm został zaimplementowany w języku Python 3.8 (program *pszt1.py*) i wykorzystuje funkcje biblioteki standardowej. Dane podawane są w postaci pliku JSON, którego format, jak i inne dodatkowe informacje, można uzyskać uruchamiając program z opcją -h. Do generowania losowych danych został napisany program *gendata.py*. Więcej informacji można uzyskać uruchamiając go z opcją -h.

## Analiza

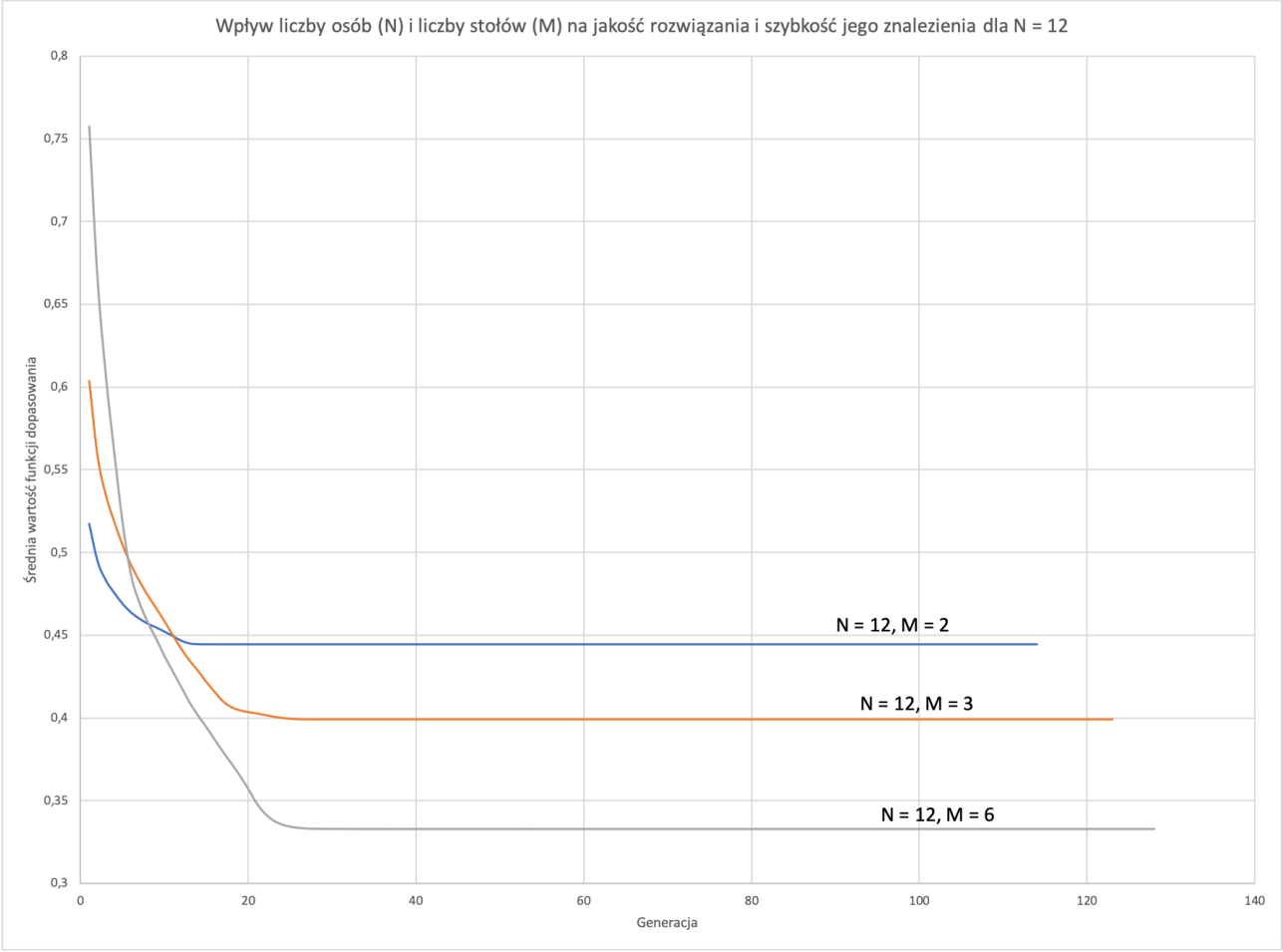
Analiza została podzielona na 3 etapy: analiza wpływu wielkości populacji na jakość rozwiązania i szybkość jego znalezienia, analiza wpływu prawdopodobieństwa mutacji na jakość znalezionego rozwiązania, analiza wpływu liczby osób i liczby stolików na szybkość znalezienia i jakość rozwiązania. Dla etapów 1 i 2 dane nie zmieniają się pomiędzy uruchomieniami, zmieniają się jedynie badane parametry. Po każdej zmianie parametrów algorytm został uruchomiony 25 razy, a wyniki uśredniono. Łącznie program został uruchomiony 550 razy. W celu przeprowadzenia analizy został stworzony program *analiza.py*. Można go wykorzystać do opracowania własnych wyników. Ze względu na dużą ilość wygenerowanych danych, zostały one umieszczone w osobnym pliku (*wyniki.xlsx*), a poniżej przedstawiono opracowane na ich podstawie wykresy. Pominięto wykres dla etapu 2 dla N = 20 w celu zachowania przejrzystości dokumentu, ale jest on dostępny w pliku *wyniki.xlsx*.



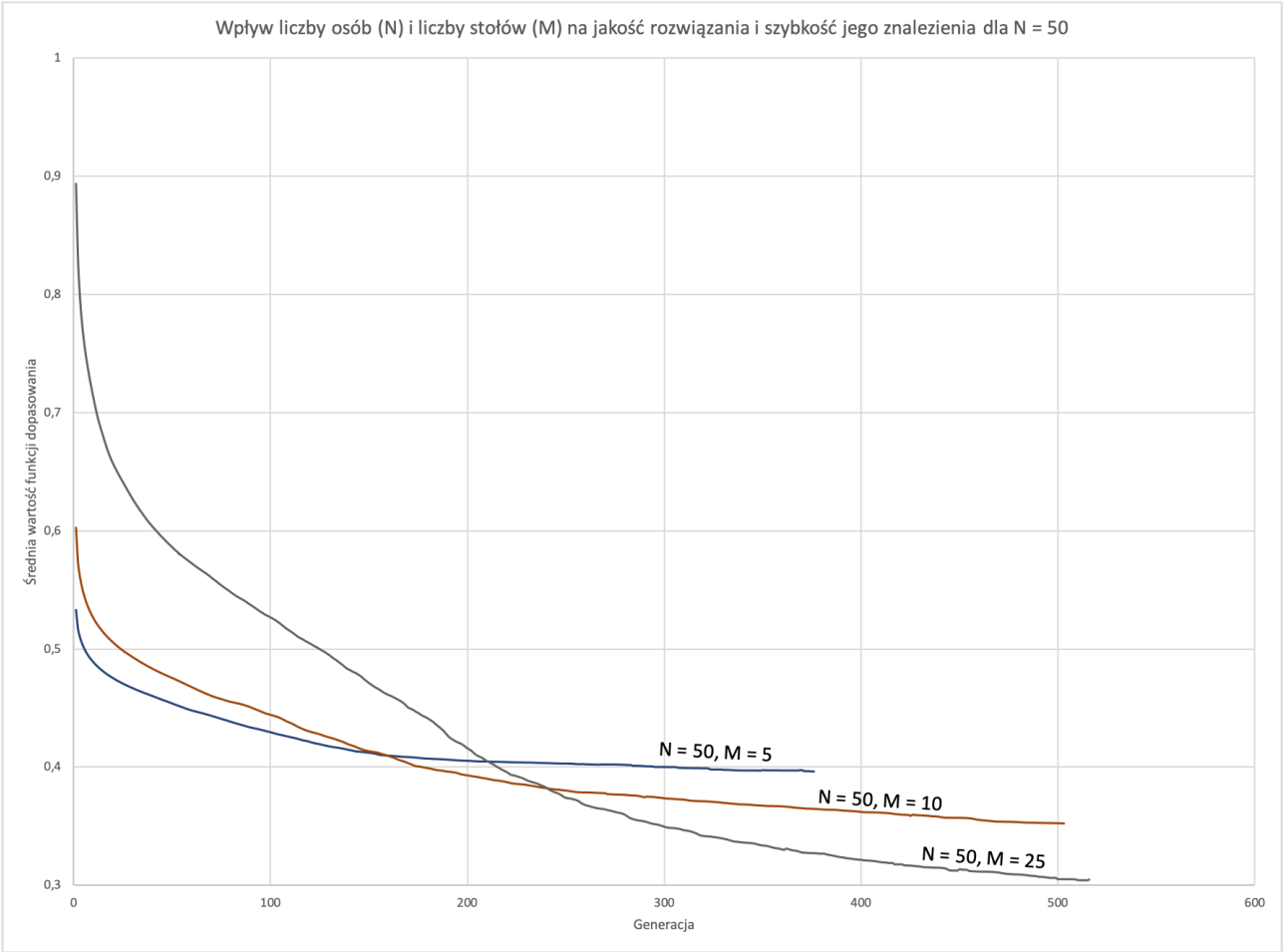

Wykres 1



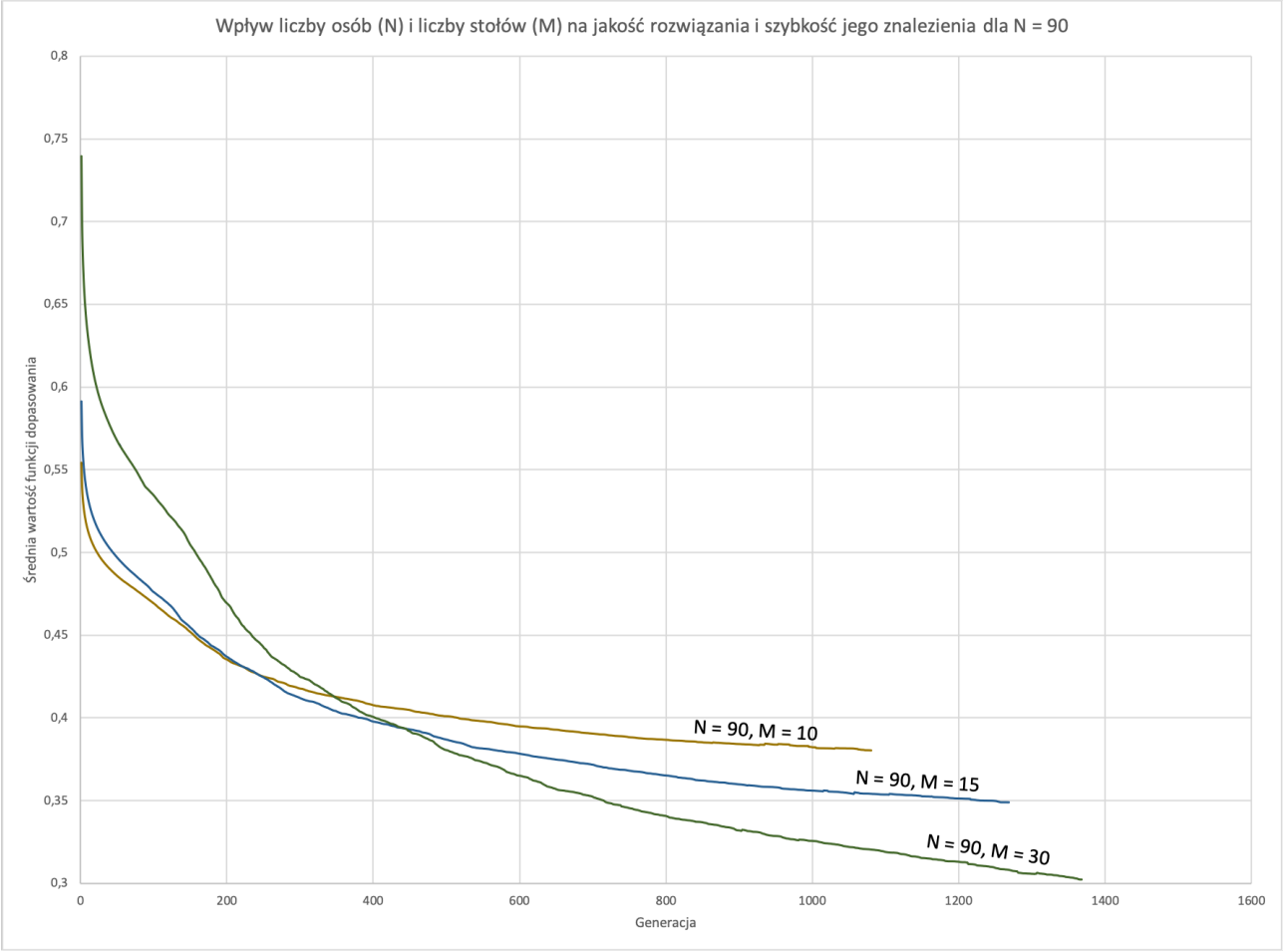

Wykres 2




Wykres 3




Wykres 4




Wykres 5

**Wykres 1:** Zbyt mały rozmiar populacji przekłada się na małą różnorodność chromosomów, a w rezultacie słabą jakość rozwiązań. Algorytm szybko znajduje minimum lokalne i nie jest w stanie się z niego wydostać. Widać to na wykresie dla $\mu = 10$. Zwiększanie liczby osobników poprawia jakość rozwiązań, ale odpowiednio wydłuża czas poszukiwań. Zestawienie pokazuje, że po przekroczeniu granicy w okolicach $\mu = 500$ jakość znalezionych rozwiązań wciąż się poprawia (ale już w mniejszym stopniu), natomiast zmniejsza się liczba generacji potrzebnych do ich znalezienia.

**Wykres 2:** Wykres wskazuje, że włączenie mutacji ($\sigma > 0$) drastycznie wpływa na działanie programu. Mutacja zwiększa właściwości eksploracyjne algorytmu, dzięki czemu znajdywane rozwiązania są dużo lepsze niż w przypadku jej braku. Poszerzenie obszaru poszukiwań znacząco jednak wydłuża czas działania. Zaskakujący jest mały wpływ prawdopodobieństwa wystąpienia mutacji na jakość rozwiązania i szybkość jego znalezienia.

**Wykresy 3, 4, 5:** Wyniki pokazują, że wraz ze zwiększaniem liczby osób algorytm potrzebuje odpowiednio więcej czasu na znalezienie rozwiązania. Zależność jest w przybliżeniu liniowa. Zwiększenie liczby dostępnych stołów korzystanie wpływa na jakość rozwiązania. Wynika to z faktu, że łatwiej znaleźć np. dwie osoby, które dość dobrze lubią się nawzajem, niż gdyby takich osób miało być dziesięć.

## Podsumowanie

Wyniki analizy są zgodne z oczekiwaniami. Większa populacja oraz włączenie mutacji (niezależnie od jej prawdopodobieństwa) pozwala uzyskać lepsze rozwiązania problemu. W praktyce jednak zbyt duża liczba osobników i zbyt duże prawdopodobieństwo mutacji wymagają większej mocy obliczeniowej komputera. Na podstawie analizy rozsądnym kompromisem mogą być wartości $\mu = 1000$ oraz $\sigma = 0.1$. Analiza danych wejściowych (tj. liczba osób i liczba stołów) pokazała, że spotykając opisywany problem w życiu codziennym warto rozważyć przygotowanie większej liczby stołów dla weselników. Pozwoli to łatwiej usadowić gości w taki sposób, aby każdy był zadowolony z towarzystwa.