Zastosowanie Algorytmów Ewolucyjnych w Rozwiazywaniu Sudoku

Jakub Pietrzak, Grzegorz Prasek

Marzec 2025

Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych
Kierunek: **Informatyka i Systemy Informacyjne**

Opiekun naukowy: dr Felicja Okulicka-Dłużewska

Marzec 2025

Contents

1 Wstep

Sudoku to jedna z najpopularniejszych łamigłówek logicznych, polegajaca na wypełnieniu siatki liczbami zgodnie z określonymi zasadami. Ze wzgledu na swoja strukture, Sudoku może być traktowane jako problem optymalizacyjny, co czyni je idealnym przypadkiem do rozwiazania przy użyciu algorytmów ewolucyjnych.

1.1 Zasady Sudoku

Plansza Sudoku składa sie z **siatki 9×9**, podzielonej na **dziewieć podkwadratów 3×3**. Zadanie polega na uzupełnieniu pustych pól cyframi od **1 do 9** tak, aby:

1. Każda cyfra wystepowała **dokładnie raz** w każdym wierszu.
2. Każda cyfra wystepowała **dokładnie raz** w każdej kolumnie.
3. Każda cyfra wystepowała **dokładnie raz** w każdym podkwadracie 3×3.

Cześć pól na planszy jest wypełniona na poczatku i nie może zostać zmieniona podczas rozwiazywania. Nasze podejście bedzie przestrzegać tych reguł, a rozwiazanie zostanie uznane za poprawne, jeśli spełni wszystkie trzy warunki.

1.2 Algorytmy ewolucyjne

Algorytmy ewolucyjne to grupa heurystycznych metod optymalizacji inspirowanych mechanizmami ewolucji biologicznej, takimi jak selekcja naturalna, mutacja i rekombinacja genów. Operuja one na populacji możliwych rozwiazań, które podlegaja iteracyjnemu ulepszaniu poprzez mechanizmy ewolucyjne. Celem jest znalezienie rozwiazania jak najlepiej dopasowanego do postawionego problemu.

W kontekście Sudoku algorytmy ewolucyjne moga służyć do generowania i poprawiania rozwiazań poprzez iteracyjne przekształcanie planszy, aż do uzyskania poprawnego układu spełniajacego zasady gry.

1.3 Algorytm genetyczny

Algorytm genetyczny (GA – *Genetic Algorithm*) to jeden z najcześciej stosowanych algorytmów ewolucyjnych. Jego działanie opiera sie na mechanizmach selekcji, krzyżowania i mutacji, które naśladuja procesy ewolucji biologicznej.

Rozwiazania problemu traktowane sa jako osobniki w populacji, a ich jakość oceniana jest na podstawie funkcji dopasowania.

W każdej iteracji algorytm wybiera najlepiej przystosowane osobniki, które nastepnie podlegaja krzyżowaniu i mutacji, tworzac nowa generacje rozwiazań. Proces ten powtarza sie, aż do osiagniecia optymalnego rozwiazania lub spełnienia warunku stopu. Algorytm genetyczny znajduje zastosowanie w wielu problemach optymalizacyjnych, w tym w rozwiazywaniu Sudoku poprzez iteracyjne modyfikowanie planszy w kierunku poprawnego układu liczb.

2 Algorytmy ewulucyjne

2.1 Podstawy algorytmów ewolucyjnych

Algorytmy ewolucyjne to grupa heurystycznych metod optymalizacji inspirowanych procesami naturalnej ewolucji. Ich działanie polega na stopniowym ulepszaniu populacji potencjalnych rozwiazań poprzez selekcje najlepiej dopasowanych osobników, ich krzyżowanie oraz mutacje. Celem tych operacji jest generowanie nowych rozwiazań, które z każda iteracja powinny coraz lepiej spełniać kryteria zadania optymalizacyjnego.

Działanie algorytmów ewolucyjnych można podzielić na kilka kluczowych etapów: inicjalizacje populacji, ocene jakości rozwiazań za pomoca funkcji dopasowania, selekcje najlepszych osobników, operacje genetyczne (krzyżowanie i mutacja) oraz tworzenie kolejnych generacji. Proces ten jest powtarzany aż do spełnienia warunku zatrzymania, np. osiagniecia rozwiazania spełniajacego wymagania problemu lub wykonania określonej liczby iteracji.

W kontekście Sudoku algorytmy ewolucyjne pozwalaja na efektywne przeszukiwanie przestrzeni możliwych układów planszy, dażac do znalezienia rozwiazania spełniajacego wszystkie reguły gry.

2.2 Wspólne etapy algorytmów przez nas rozważanych

2.2.1 DNA z sudoku

Nasz algorytm wykorzystuje DNA, czyli ciagi liczb odpowiadajace pustym polom w planszy Sudoku.

Dla danej planszy Sudoku:

5		**4**	**6**	**7**	**8**	**9**	**1**	**2**
6	**7**		**1**	**9**	**5**	**3**	**4**	**8**
1	**9**	**8**	**3**	**4**	**2**	**5**	**6**	
8	**5**	**9**	**7**	**6**	**1**		**2**	**3**
4	**2**	**6**	**8**	**5**	**3**	**7**	**9**	**1**
	1	**3**	**9**	**2**	**4**	**8**	**5**	**6**
9	**6**		**5**	**3**	**7**	**2**	**8**	**4**
2	**8**	**7**	**4**	**1**	**9**	**6**	**3**	**5**
3	**4**	**5**	**2**	**8**	**6**	**1**	**7**	**9**

algorytm wygeneruje losowy ciag liczb, np. 234529, który bedzie odpowiadał wartościom w pustych polach planszy.

5	2	**4**	**6**	**7**	**8**	**9**	**1**	**2**
6	**7**	3	**1**	**9**	**5**	**3**	**4**	**8**
1	**9**	**8**	**3**	**4**	**2**	**5**	**6**	4
8	**5**	**9**	**7**	**6**	**1**	5	**2**	**3**
4	**2**	**6**	**8**	**5**	**3**	**7**	**9**	**1**
2	**1**	**3**	**9**	**2**	**4**	**8**	**5**	**6**
9	**6**	9	**5**	**3**	**7**	**2**	**8**	**4**
2	**8**	**7**	**4**	**1**	**9**	**6**	**3**	**5**
3	**4**	**5**	**2**	**8**	**6**	**1**	**7**	**9**

2.2.2 Funkcja dopasowania

Funkcja dopasowania (*fitness function*) jest kluczowym elementem algorytmów ewolucyjnych, ponieważ określa jakość poszczególnych rozwiazań w populacji. W przypadku Sudoku jej zadaniem jest mierzenie stopnia poprawności danego układu liczb na planszy.

W naszym podejściu funkcja dopasowania oblicza sume wszystkich błedów w rozwiazaniu, tj. liczbe powtórzeń w poszczególnych sekcjach planszy. Definiujemy ja jako:

fitness = liczba powtórzeń w wierszach + liczba powtórzeń w kolumnach + liczba powtórzeń w kwadratach 3×3

Gdzie:

- **Powtórzenie w wierszu** oznacza, że dana cyfra pojawia sie wiecej niż raz w tym samym wierszu.
- **Powtórzenie w kolumnie** oznacza, że dana cyfra wystepuje wiecej niż raz w tej samej kolumnie.
- **Powtórzenie w kwadracie 3×3** oznacza, że cyfra powtarza sie w obrebie jednego z dziewieciu podobszarów siatki Sudoku.

Optymalne rozwiazanie, czyli poprawnie wypełniona plansza Sudoku, ma wartość $fitness = 0$, co oznacza brak powtórzeń. Im wyższa wartość funkcji dopasowania, tym gorsze rozwiazanie, co wskazuje na wieksza liczbe błedów

w układzie liczb. Algorytm ewolucyjny daży do minimalizacji tej wartości poprzez kolejne operacje selekcji, krzyżowania i mutacji, stopniowo eliminujac błedne konfiguracje planszy.

2.2.3 Selekcja

Selekcja to proces wyboru osobników, które zostana wykorzystane do tworzenia nowego pokolenia w algorytmie ewolucyjnym. Jej celem jest preferowanie lepszych rozwiazań, które maja mniejsza liczbe powtórzeń cyfr w wierszach, kolumnach i podkwadratach 3×3. Dzieki temu populacja stopniowo ewoluuje w kierunku poprawnego rozwiazania Sudoku.

W algorytmach ewolucyjnych stosuje sie różne metody selekcji, spośród których najpopularniejsze to:

- **Selekcja ruletkowa** – prawdopodobieństwo wyboru osobnika jest proporcjonalne do jego dopasowania. Może prowadzić do dominacji jednej jednostki, jeśli różnice w dopasowaniu sa zbyt duże.
- **Selekcja turniejowa** – losujemy kilku osobników z populacji, a najlepszy z nich przechodzi do kolejnej generacji. Pozwala kontrolować intensywność selekcji i zachować różnorodność rozwiazań.
- **Selekcja rankingowa** – szanse na wybór sa uzależnione od pozycji w rankingu populacji, co pozwala uniknać dominacji jednej jednostki.
- **Selekcja elitarna** – najlepsze rozwiazania sa bezpośrednio przenoszone do nowej populacji, dzieki czemu nie tracimy najlepszego dotychczasowego rozwiazania.

W kontekście Sudoku planujemy zaimplementować wszystkie wymienione metody selekcji, a nastepnie porównać ich skuteczność pod katem szybkości konwergencji oraz jakości uzyskiwanych rozwiazań. Celem tego porównania jest wybór optymalnej metody selekcji, która najlepiej sprawdzi sie w procesie ewolucji populacji plansz Sudoku, minimalizujac liczbe błedów i maksymalizujac efektywność algorytmu.

2.3 Algorytm genetyczny

Algorytm genetyczny czerpie inspiracje z natury, gdzie najlepiej przystosowane osobniki maja najwieksza szanse na przeżycie i reprodukcje. Działa on na podobnej zasadzie – ocenia kandydatów (rozwiazania) według funkcji dopasowania, a nastepnie selekcjonuje najlepsze osobniki, które sa poddawane procesowi krzyżowania i mutacji, aby uzyskać nowe rozwiazania.

W naszym przypadku algorytm jako dane wejściowe otrzymuje plansze Sudoku z cześciowo uzupełnionymi liczbami. Reprezentacja rozwiazania (DNA) bedzie ciagiem liczb odpowiadajacym pustym polom na planszy. Celem algorytmu jest znalezienie takiego DNA, które po wstawieniu do odpowiednich miejsc na planszy spełni wszystkie zasady Sudoku.

2.3.1 Przebieg algorytmu

1. **Inicjalizacja populacji** – na poczatku generujemy N losowych ciagów liczb o długości równej liczbie pustych pól na planszy.

2. **Ocena dopasowania** – dla każdego osobnika (ciagu) obliczamy wartość funkcji dopasowania, określajaca zgodność rozwiazania z zasadami Sudoku.

3. **Selekcja najlepszych** – wybieramy M najlepiej dopasowanych ciagów liczb do kolejnego etapu.

4. **Krzyżowanie** – łaczymy wybrane ciagi, tworzac nowe osobniki poprzez losowe wymieszanie fragmentów dwóch rodziców.

5. **Mutacja** – wprowadzamy losowe zmiany w wartościach niektórych liczb w nowych ciagach.

6. **Sprawdzenie rozwiazania** – jeśli któryś z nowych osobników spełnia wszystkie warunki Sudoku, kończymy algorytm. W przeciwnym razie ponawiamy kroki od 2 do 5 aż do znalezienia poprawnego rozwiazania.

2.3.2 Krzyżowanie

Krzyżowanie polega na podziale dwóch ciagów w losowych miejscach na P fragmentów i ich naprzemiennym łaczeniu. Przykładowo, dla dwóch ciagów:

462154378932147 i 868346125451265

możemy podzielić je w nastepujacy sposób:

46—2154—37893—2147 oraz 86—8346—12545—1265

Po naprzemiennym połaczeniu otrzymujemy:

46—8346—37893—1265

2.3.3 Mutacja

Mutacja polega na losowej zmianie wybranych wartości w ciagu. Przykładowo, jeśli mamy ciag:

$$46215437893214 7$$

to po mutacji może on przyjać postać:

$$49215417893214 7$$

Zmiana wartości wprowadza element losowości, który pozwala uniknać lokalnych minimów i zwieksza szanse na znalezienie poprawnego rozwiazania.

2.4 Algorytm ewolucyjny

Programowanie ewolucyjne, podobnie jak algorytm genetyczny, czerpie inspiracje z natury. Jednak w przeciwieństwie do algorytmów genetycznych, które kłada duży nacisk na krzyżowanie, programowanie ewolucyjne skupia sie przede wszystkim na mutacjach, adaptacji i selekcji.

Podobnie jak w algorytmie genetycznym, na wejściu otrzymujemy plansze Sudoku, a osobniki populacji reprezentowane sa przez ciagi liczb odpowiadajace pustym polom w planszy.

2.4.1 Przebieg algorytmu

1. **Inicjalizacja populacji** – Generujemy N losowych ciagów liczb odpowiadajacych pustym polom w planszy.

2. **Ocena dopasowania** – Każdy osobnik jest oceniany według funkcji dopasowania, a nastepnie wybierane jest M najlepszych ciagów.

3. **Adaptacja** – W analizie każdego osobnika identyfikujemy powtarzajace sie cyfry w wierszach, kolumnach i sektorach 3×3. Nastepnie zamieniamy je na mniej powtarzajace sie wartości, co zwieksza szanse na poprawne rozwiazanie.

4. **Mutacja** – Na nowej populacji losowo zmieniamy wartości w ciagach, aby zapewnić wieksza różnorodność.

5. **Sprawdzenie rozwiazania** – Jeśli któryś z osobników poprawnie rozwiazuje Sudoku, algorytm kończy działanie. W przeciwnym razie powtarzamy powyższe kroki.

2.4.2 Adaptacja

Podczas procesu adaptacji analizujemy DNA osobników i wykrywamy cyfry, które powtarzaja sie w tym samym wierszu, kolumnie lub sektorze 3×3. Nastepnie zastepujemy je innymi cyframi w taki sposób, aby zmniejszyć liczbe kolizji. Ten krok znaczaco przyspiesza algorytm w kierunku poprawnego rozwiazania.

Poniżej przedstawiono przykładowa plansze Sudoku, w której niepoprawnie powtórzyły sie cyfry (oznaczone na czerwono):

5	**3**	7	6	**7**	8	9	1	2
6	7	2	**1**	**9**	**5**	3	4	8
7	**9**	**8**	3	4	2	5	**6**	7
8	5	9	**8**	6	**3**	4	2	**1**
7	6	1	2	9	4	8	5	**6**
4	2	3	**6**	5	**7**	1	9	8
9	**6**	1	5	3	7	**2**	**8**	4
2	8	7	**4**	**1**	**9**	6	3	**5**
3	4	5	8	**8**	6	7	**7**	**9**

W tym przypadku cyfry **7** (oznaczone na czerwono) powtarzaja sie w sektorze 3×3, co jest niepoprawne. Algorytm automatycznie wykryje te kolizje i zamieni powtarzajace sie wartości na inne. Po przeprowadzeniu procesu adaptacji otrzymujemy poprawiona plansze:

5	**3**	4	6	**7**	8	9	1	2
6	7	2	**1**	**9**	**5**	3	4	8
1	**9**	**8**	3	4	2	5	**6**	7
8	5	9	**8**	6	**3**	4	2	**1**
7	6	1	2	9	4	8	5	**6**
4	2	3	**6**	5	**7**	1	9	8
9	**6**	1	5	3	7	**2**	**8**	4
2	8	7	**4**	**1**	**9**	6	3	**5**
3	4	5	8	**8**	6	7	**7**	**9**

Dzieki temu algorytm skutecznie eliminuje błedy i przyspiesza dochodzenie do poprawnego rozwiazania Sudoku.

3 Parametry programu

Aby algorytm ewolucyjny mógł skutecznie działać i znajdować poprawne rozwiazania Sudoku, konieczne jest odpowiednie dobranie jego parametrów. Wartości te maja kluczowy wpływ na przebieg ewolucji, szybkość konwergencji oraz jakość otrzymywanych rozwiazań. Nieodpowiednie ustawienia moga prowadzić do zbyt wolnego postepu lub do utkniecia w lokalnych minimach.

Parametry, które zostana dostosowane i przetestowane w kontekście naszego algorytmu, obejmuja:

- **Rozmiar populacji** – liczba osobników w każdej generacji,
- **Maksymalna liczba iteracji** – maksymalna liczba generacji, po której algorytm zostanie zatrzymany,
- **Współczynnik mutacji** – prawdopodobieństwo wystapienia mutacji w nowo powstałych osobnikach,
- **Kryteria zatrzymania** – warunki decydujace o zakończeniu działania algorytmu,
- **Ilość potomków tworzonych z dwóch rodziców** – liczba nowych osobników generowanych w procesie krzyżowania.
- **Ilość cześci dzielenia DNA** – na ile cześci bedzie dzielone DNA podczas krzyżowania.

Odpowiednie dobranie tych parametrów wymaga eksperymentów oraz porównania różnych konfiguracji pod katem skuteczności algorytmu.

3.1 Rozmiar populacji

Rozmiar populacji określa, ile osobników bedzie uczestniczyć w każdej iteracji algorytmu ewolucyjnego. Wieksza populacja zwieksza różnorodność genetyczna, co może poprawić jakość końcowego rozwiazania, ale jednocześnie wydłuża czas obliczeń.

Zbyt mała populacja może prowadzić do szybkiej konwergencji do lokalnego optimum, ponieważ algorytm nie bedzie miał wystarczajacej liczby wariantów do eksploracji przestrzeni rozwiazań. Z drugiej strony, zbyt duża populacja zwieksza koszt obliczeniowy każdej generacji i może znaczaco spowolnić działanie algorytmu.

Dla naszego algorytmu planujemy przeprowadzić testy z różnymi wartościami populacji, zaczynajac od wartości $N = 50$ i stopniowo zwiekszajac ja do

$N = 500$, aby znaleźć optymalny balans miedzy różnorodnościa a szybkościa działania.

3.2 Współczynnik mutacji

Mutacja to kluczowy mechanizm algorytmów ewolucyjnych, który wprowadza losowe zmiany w populacji, zapobiegajac przedwczesnej konwergencji do lokalnego optimum. W kontekście Sudoku mutacja polega na zamianie losowych cyfr w pustych polach planszy, przy jednoczesnym zachowaniu reguł gry.

Współczynnik mutacji określa prawdopodobieństwo, z jakim mutacja wystepuje w nowo powstałych osobnikach. Zbyt niski współczynnik może prowadzić do stagnacji algorytmu i utraty różnorodności populacji, natomiast zbyt wysoki może powodować utrate korzystnych cech najlepszych rozwiazań.

Dla naszej implementacji planujemy testować różne wartości współczynnika mutacji, poczawszy od **1%** do **10%**, aby znaleźć optymalna wartość pozwalajaca na skuteczne poszukiwanie rozwiazania przy zachowaniu stabilnej ewolucji populacji.

3.3 Ilość dzieci z dwóch rodziców

W algorytmach genetycznych liczba dzieci tworzonych z pary rodziców wpływa na tempo ewolucji populacji. W klasycznych implementacjach najcześciej stosuje sie jedno lub dwoje potomków na pare, ale w niektórych przypadkach możliwe jest generowanie wiekszej liczby dzieci.

Wieksza liczba dzieci pozwala na szybsza eksploracje przestrzeni rozwiazań, ale jednocześnie zwieksza obciażenie obliczeniowe. Zbyt duża liczba potomków może prowadzić do nadmiernej zmienności i utraty dobrych cech rozwiazań, podczas gdy zbyt mała liczba może spowolnić proces optymalizacji.

Dla Sudoku planujemy przetestować różne warianty, w tym **1, 2 oraz 3 dzieci na pare rodziców**, aby określić optymalna strategie dla naszego algorytmu.

3.4 Kryteria zatrzymania

Kryteria zatrzymania określaja warunki, po spełnieniu których algorytm ewolucyjny kończy swoje działanie. Ich dobór jest kluczowy dla zapewnienia efektywności obliczeniowej oraz unikniecia niepotrzebnych iteracji w przypadku znalezienia poprawnego rozwiazania.

Najcześciej stosowane kryteria zatrzymania obejmuja:

- **Maksymalna liczbe iteracji** – algorytm kończy działanie po osiagnieciu określonej liczby generacji.
- **Znalezienie idealnego rozwiazania** – jeśli w populacji pojawi sie osobnik o funkcji dopasowania równej 0, oznacza to poprawnie rozwiazane Sudoku i algorytm może zostać zatrzymany.

3.4.1 Maksymalna liczba iteracji

Ustalenie maksymalnej liczby iteracji jest istotne dla zapewnienia, że algorytm nie bedzie działał w nieskończoność w przypadku, gdy nie może znaleźć poprawnego rozwiazania. Zbyt niska wartość może uniemożliwić dotarcie do optymalnego wyniku, a zbyt wysoka prowadzi do zbednego wydłużania obliczeń.

Planowane testy obejma wartości od 100 do 5000 iteracji, aby określić, po ilu generacjach algorytm osiaga najlepsze wyniki w kontekście Sudoku.

3.4.2 Znalezienie idealnego rozwiazania

Najbardziej efektywnym sposobem zakończenia algorytmu jest wykrycie osobnika, którego funkcja dopasowania wynosi 0, co oznacza, że rozwiazanie Sudoku zostało poprawnie wygenerowane. W takim przypadku dalsze iteracje staja sie zbedne i algorytm może zostać natychmiast zatrzymany.

3.5 Ilość cześci dzielenia DNA

Podczas krzyżowania DNA należy je podzielić w taki sposób, aby potomek składał sie z fragmentów DNA obojga rodziców. Parametr ten określa, na ile cześci należy podzielić DNA przed krzyżowaniem.