Rozwiązywanie Sudoku przy pomocy algorytmu genetycznego oraz algorytmu ACO

Bartosz Cywiński, Łukasz Staniszewski

1 Opis problemu

Podstawowa wersja Sudoku składa się z dwuwymiarowej planszy 9x9, podzielonej na 9 pól 3x3. Planszę Sudoku da się jednak rozszerzać do większych rozmiarów. Plansza na początku gry częściowo uzupełniona jest liczbami. Początkowy rozkład liczb ma duże znaczenie dla dalszego przebiegu gry – jest on jedynym czynnikiem decydującym o trudności danej wersji Sudoku. Celem rozwiązania łamigłówki jest takie ułożenie cyfr od 1 do 9, aby w każdym rzędzie, każdej kolumnie i każdym polu 3x3 każda cyfra występowała dokładnie jeden raz. Zostało udowodnione, że problem rozwiązania Sudoku jest problemem NP-trudnym, dlatego też jest on dobrym testem efektywności algorytmów.

2 Opis rozwiązania przy pomocy algorytmu ACO

2.1 Pseudokod i opis algorytmu

W zaproponowanym rozwiązaniu użyty został algorytm Ant Colony System (ACS), będący ulepszoną wersją klasycznego algorytmu Ant System (AS). Jego cechą charakterystyczną jest zastosowanie lokalnych aktualizacji feromonów przez każdą mrówkę oraz to, że tylko najlepsza mrówka aktualizuje globalne wartości tablicy feromonów. Ponadto po każdym wybraniu nowego elementu rozwiązania (wpisaniu cyfry w wybraną komórkę planszy) aktualizowane zostają wszystkie wartości możliwe do przyjęcia przez komórki planszy oraz spełniające zasady łamigłówki, ponieważ na skutek wybrania elementu rozwiązania mogły się one zmienić.

Linie 2-4: Każda komórka o indeksie i na początku algorytmu posiada taki sam zbiór możliwych wartości do przyjęcia ($\mathbb{V}_i = \{1, ..., 9\}$). Aktualizacje ograniczeń komórek z wybraną wartością polegają na:

- Eliminacji z $\mathbb{V}_i$ wartości wybranych przez sąsiadów komórki i, gdzie sąsiedzi to wszystkie komórki w tym samym rzędzie, tej samej kolumnie oraz tym samym polu 3x3,
- Jeśli $|\mathbb{V}_i| = 1$, to przypisz wartość do komórki.

Linia 5: Dla Sudoku o wymiarze d definiujemy dwuwymiarową globalną tablicę feromonów τ, w której każdy element jest oznaczony jako τ_i^k, gdzie i to indeks komórki planszy ($0 \leq i \leq d^2 - 1$), a k to możliwa wartość do przyjęcia przez komórkę ($k \in [1, d]$). Każdy element tablicy feromonów inicjowany jest wartością $\tau_0 = \frac{1}{c}$, gdzie $c = d^2$ jest całkowitą liczbą komórek planszy. W tablicy trzymane są wartości feromonów każdej możliwej do przyjęcia wartości w każdej komórce planszy.

Algorithm 1 Algorytm ACS do rozwiązywania Sudoku

```
1: wczytaj planszę Sudoku
2: for komórka z wybraną wartością do
3:    aktualizuj możliwe do przyjęcia wartości przez sąsiadów tej komórki
4: end for
5: zainicjuj tablicę feromonów
6: while !stop do
7:    daj każdej mrówce kopię planszy Sudoku
8:    for liczba komórek do
9:       for liczba mrówek do
10:         if komórka nie ma wybranej wartości then
11:            wybierz wartość dla komórki
12:            aktualizuj ograniczenia
13:            aktualizuj lokalny feromon
14:         end if
15:      end for
16:   end for
17:   znajdź najlepszą mrówkę
18:   aktualizuj tablicę feromonów
19:   parowanie najlepszej wartości
20: end while
```

Linie 8-10: Zbiór m mrówek buduje częściowe rozwiązanie s^p ze skończonego zbioru możliwych elementów rozwiązania $\mathbf{C} = \{c_i^k\},\ i = 0,...,d^2-1,\ k \in [1,d]$, gdzie i to indeks komórki planszy, a k to możliwa cyfra do wpisania w komórkę. Każda mrówka zaczyna od losowo wybranej komórki planszy Sudoku. Przechodzi ona po każdej możliwej komórce na planszy. Za każdym razem, kiedy mrówka natrafia na komórkę, która nie ma przypisanej wartości, podejmuje ona decyzję jaka to powinna być wartość i ją przypisuje - do s^p dodawany jest element rozwiązania ze zbioru $\mathbf{N}(s^p) \subseteq \mathbf{C}$, który jest zdefiniowany jako zbiór elementów, które mogą być dodane do częściowego rozwiązania s^p bez naruszenia założonych ograniczeń.

Linia 11: Wybór wartości dla komórki i dokonywany jest za pomocą stochastycznego mechanizmu uwzględniającego wartości w tablicy feromonów. Zdefiniujmy zbiór $\mathbb{V}_i$, jako zbiór możliwych do przyjęcia wartości przez komórkę i. Istnieją do wyboru dwie metody:

- Selekcja zachłanna, gdzie wybieramy element ze zbioru $\mathbb{V}_i$ z najwyższą wartością w tablicy feromonów,
- Selekcja ruletkowa, gdzie prawdopodobieństwa są proporcjonalne do wartości w tablicy feromonów.

Prawdopodobieństwa wybierane są na podstawie parametru $q_0 \in [0,1]$. Kolejny element rozwiązania s jest zatem wybierany według wzoru:

$$s = \begin{cases} \arg\max_{k \in \mathbb{V}_i} \tau_i^k, & \text{jeśli } q \leq q_0, \\ \text{R}, & \text{w pozostałych przypadkach,} \end{cases} \tag{2.1}$$

gdzie wartość $q \in [0,1]$ losowana jest z rozkładem jednostajnym, natomiast R jest określona jako:

$$p_i^k = \frac{\tau_i^k}{\sum_{j \in \mathbb{V}_i} \tau_i^j}, k \in \mathbb{V}_i \tag{2.2}$$

gdzie p_i^k to prawdopodobieństwo wyboru k z $\mathbb{V}_i$ na miejscu o indeksie i.

Linia 13: Aktualizowanie lokalnego feromonu wspomaga eksplorację przestrzeni przeszukiwań. Za każdym razem kiedy wartość s zostaje przypisana do komórki w planszy o indeksie i, aktualizacja następuje według formuły:

$$\tau_i^s = (1-\varphi)\tau_i^s + \varphi\tau_0, \tag{2.3}$$

gdzie $\varphi \in (0,1]$ to parametr rozpadu feromonu, który jest zazwyczaj mały, a τ_0 jest początkową wartością feromonu.

Celem aktualizacji tablicy feromonów jest zwiększenie wartości feromonów związanych z dobrymi rozwiązaniami i zmniejszenie wartości związanych z gorszymi rozwiązaniami.

Linie 17-18: Aby znaleźć najlepsze rozwiązanie, należy znaleźć najlepszą mrówkę w iteracji, czyli taką, która uzupełniła najwięcej komórek w planszy Sudoku w danej iteracji algorytmu. Aktualizacja tablicy feromonów następuje według wzoru:

$$\tau_i^s = \begin{cases} (1-\rho)\tau_i^s + \rho\Delta\tau_i^s, & \text{jeśli najlepsze rozwiązanie w iteracji jest najlepsze globalnie,} \\ \tau_i^s, & \text{w pozostałych przypadkach,} \end{cases} \tag{2.4}$$

gdzie $\rho \in [0,1]$ to parametr parowania feromonu, a $\Delta\tau_i^s$ można przyjąć jako:

$$\Delta\tau_i^s = \frac{c}{c - f_{best}}, \tag{2.5}$$

gdzie f_{best} to liczba komórek uzupełnionych przez najlepszą mrówkę, a c to całkowita liczba komórek na planszy.

Linia 19: Na koniec stosowane jest parowanie najlepszej wartości, które zapobiega przedwczesnej zbieżności algorytmu:

$$\Delta\tau_{best} = \Delta\tau_{best}(1-\rho_{best}), \tag{2.6}$$

gdzie ρ_{best} to parametr parowania feromonu najlepszej wartości.

3 Opis rozwiązania przy pomocy algorytmu genetycznego

3.1 Opis algorytmu

Przed zdefiniowaniem samego zastosowania algorytmu do problemu, dobrze jest określić części na jakie składa się Algorytm Genetyczny. W typowym takim algorytmie zaczyna się od inicjalizacji populacji startowej - jest to populacja, z którą zaczynamy. Następnie, algorytm opiera się na pętli, która trwa do momentu spełnienia warunku stopu. W pętli tej pierwszym krokiem jest ocenienie całej aktualnej populacji przy użyciu funkcji dopasowania i posortowanie populacji rosnąco względem tego wskaźnika. Następnie wykonywana jest selekcja, polegająca na wybraniu z aktualnej populacji tych osobników, którzy mają przejść do kolejnej populacji, pozostałych się odrzuca. Kolejno, do momentu aż rozmiar nowej populacji nie pokryje się z rozmiarem poprzedniej, wybierani są z wyselekcjonowanej części dwaj rodzice ze zwracaniem, na nich wykonywane jest krzyżowanie (gdzie mieszamy informację o jednym rodzicu z informacją o drugim rodzicu), w wyniku czego otrzymujemy dwoje dzieci, każdy zawierający inne części swoich rodziców. Na takich dzieciach wykonywana jest mutacja, polegająca na losowej zamianie elementów w chromosomie.

W algorytmie tym mamy do czynienia z hiperparametrami - jest to między innymi rozmiar populacji, prawdopodobieństwo krzyżowania czy prawdopodobieństwo mutacji. Również, w ramach algorytmów genetycznych, konieczne jest zdefiniowanie funkcji dopasowania pasującej do problemu, metody selekcji nowej populacji, czy sposobie krzyżowania i mutowania dzieci.

3.2 Zastosowanie algorytmu do Sudoku

Na początku zdefiniujmy sam chromosom - będzie to ciąg 81 liczb z zakresu 1-9, podzielonych na 9 podciągów, reprezentujących jeden podblok sudoku 3x3. Do algorytmu wprowadzony jest również ciąg pomocniczy - definiuje on stałe, niezmienne liczby w chromosomie - składa się on zarówno z liczb z zakresu 1-9 definiujących stałe miejsca w planszy oraz liczb 0 - oznaczających miejsce, które może być w wyniku kolejnych generacji zmieniane w chromosomie.

Kolejnym elementem do zdefiniowania jest warunek stopu - zakładamy tutaj, że generowanie kolejnych populacji jest zakończone dopiero, gdy całe sudoku zostanie rozwiązane. Ze względu na możliwość utykania przez sudoku w optimach lokalnych, dobrym pomysłem może okazać się zastosowanie dodatkowego elementu w algorytmie - jeśli liczba iteracji w ramach których się nie poprawił wynik będzie zbyt duża, można przeprowadzić reinicjalizację populacji poprzez zachowanie małej części obecnej populacji i dopełnienie jej o nowe, losowo wygenerowane chromosomy.

Sudoku swoim charakterem przypomina problem optymalizacji ze spełnianiem ograniczeń - aby rozwiązanie było poprawne musi ono spełniać następujące warunki:

- ciąg elementów w każdej kolumnie sudoku musi być permutacją ciągu [1 2 3 4 5 6 7 8 9],
- ciąg elementów w każdym wierszu sudoku musi być permutacją ciągu [1 2 3 4 5 6 7 8 9],
- ciąg elementów w każdym podbloku 3x3 musi być permutacją ciągu [1 2 3 4 5 6 7 8 9],
- w wynikowej planszy nie może nastąpić zmiana liczby w miejscach, które zostały podane na wejściu.

Zauważmy, jednak, że jeśli odpowiednio zajmiemy się losowaniem populacji startowej, mutacją i krzyżowaniem, warunek z ciągiem elementów w każdym podbloku będzie zawsze spełniany. Dodatkowo, jeśli wprowadzimy podany wcześniej ciąg pomocniczy, ostatni z tych warunków też zapewnimy - w ten sposób dla funkcji celu mamy do spełnienia 18 ograniczeń (osobno na każdy wiersz i osobno na każdą kolumnę).

Losowanie populacji startowej będzie odbywało się oddzielnie w ramach każdego podciągu każdego chromosomu - w wolnych miejscach w podciągu losowo rozmieścimy wszystkie pozostałe do rozłożenia liczby. Wtedy mamy zapewnione, że w populacji startowej, w każdym chromosomie nie ma naruszenia warunku z podblokiem 3x3.

Selekcja jaką zastosujemy do zachowania części populacji to selekcja elitarna ze współczynnikiem elitarności wynoszącym 2 (co oznacza, że 2 najlepsze chromosomy z poprzedniej populacji pozostaną niezmienione w następnej), a wybór rodziców do krzyżowania będzie wykonywany przy użyciu selekcji turniejowej.

Krzyżowanie jakie zostanie zastosowane to krzyżowanie jednorodne, będzie wykonywane tylko między podblokami, tak więc dla ciągu 81 elementów możliwych punktów krzyżujących jest 8.

Mutacje, wykonywane z odpowiednim prawdopodobieństwem, będą miały miejsce tylko wewnątrz danego podbloku (w ramach podciągu wielkości 9 elementów), zastosujemy w tym przypadku mutację wymieniającą (zastępującą liczby miejscami), jednak gdy wymiana naruszy pozycje liczb w sto-

sunku do ciągu wejściowego, mutacja będzie porzucana.

Po pojedynczym krzyżowaniu i zastosowaniu mutacji na wynikach krzyżowania otrzymujemy dwa nowe chromosomy, będące częścią nowej populacji. Warto zwrócić uwagę na fakt, że te dwie operacje, zastosowane w powyższy sposób, nie naruszają zarówno warunku z prawidłowym ciągiem elementów w podbloku 3x3 Sudoku, jak i nie zmieniają liczb nienaruszalnych z punktu widzenia planszy wejściowej.

Ostatnim elementem pozostałym do zdefiniowania jest funkcja dopasowania f, którą chcemy minimalizować. Optymalne rozwiązanie (pełne rozwiązanie Sudoku) oznacza znalezienie chromosomu, dla którego $f = 0$. Funkcja, jaką proponujemy, polega na zainicjowaniu $f \leftarrow 0$ i przejściu po wszystkich wierszach i kolumnach Sudoku i dodaniu kary w postaci ilości nieznajdujących się w ramach wiersza lub kolumny liczb ze zbioru $\{1,2,3,4,5,6,7,8,9\}$. Dodatkowym pomysłem jest dodanie do wartości funkcji $f \leftarrow f + i$, gdy najlepszy wynik z aktualnej populacji jest nie lepszy niż najlepszy z poprzednich i epok, co ułatwić może uciekanie od optimów lokalnych.

4 Testy numeryczne

- Porównanie efektywności algorytmu ACO dla różnej liczby mrówek.
- Porównanie efektywności algorytmów dla plansz Sudoku o różnym poziomie trudności (przy czym trudność plansz oceniana będzie na podstawie literatury).
- Porównanie efektywności algorytmów dla różnych wartości hiperparametrów, takich jak prawdopodobieństwa mutacji i krzyżowania czy rozmiar populacji.
- Porównanie czasu potrzebnego algorytmom do znalezienia rozwiązania.
- Porównanie liczby epok i wywołań funkcji dopasowania potrzebnych do znalezienia rozwiązania w algorytmie genetycznym.
- Przy testowaniu efektywności algorytmów będą one uruchamiane kilkanaście razy - będziemy mierzyć średnią, medianę, odchylenie standardowe wyników.

5 Wybrane technologie

Algorytmy zaimplementowane zostaną w języku Python. Do wizualizacji wyników i porównywania efektywności algorytmów wykorzystamy bibliotekę Matplotlib. Klasy potrzebne do implementacji wykonane zostaną jako skrypty Python'owe, natomiast wizualizacja, porównywanie i testowanie algorytmów wykonamy w Jupyter Notebook'ach - dla czytelności odczytu i wygody wykonywania wielu testów przy jednoczesnej ich wizualizacji na wykresach.

6 Bibliografia

[1] Mirjalili, S. (2019). "Evolutionary Algorithms and Neural Networks: Theory and Applications". Griffith University Brisbane, QLD: Springer International Publishing.
[2] Lloyd, H. and Amos, M. (2020). "Solving Sudoku With Ant Colony Optimization". IEEE.
[3] Dorigo, M., Birattari, M. and Stutzle, T. "Ant Colony Optimization" J. Autom., IEEE, 2006.
[4] T. Mantere and J. Koljonen, "Solving, rating and generating Sudoku puzzles with GA," 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 1382-1389, doi: 10.1109/CEC.2007.4424632.