Sztuczna Inteligencja

Problemy spełniania ograniczeń

Weronika Babicz 234797

Spis treści

1 Implementacja

Zaimplementowano dwa podstawowe algorytmy rozwiązywania problemów spełniania ograniczeń (CSP) i zastosowano je do rozwiązania popularnej łamigłówki - Sudoku.

1.1 Algorytm przeszukiwania z nawrotami

Algorytm ten zakłada systematyczne rozwiązywanie problemu - stopniowo wypełniane są poszczególne pola Sudoku, a w przypadku, gdy przypisanie jest niedozwolone z zasadami gry (niezgodne z ograniczeniami), przypisanie zostaje wycofane, a prawidłowej wartości szuka się dalej od bieżącego miejsca.

Takie rozwiązanie przypomina strukturę drzewa - każdy węzeł jest kolejnym wypełnieniem jednego pola Sudoku, a nawrót oznacza przejście w górę drzewa.

1.2 Algorytm przeszukiwania w przód

Algorytm ten również działa na strukturze drzewa, jednakże w momencie dodawania węzła (czyli wypełniania pola Sudoku) sprawdza się, czy przy próbie wypełnienia kolejnego pola nie pozbawi się możliwości wybrania wartości dla pól, których dziedziny uległy zmianie na rzecz zmienionego wcześniej pola.

To znaczy, że jeżeli dodając węzeł natknie się na sytuację, w której dodanie kolejnego węzła w tym poddrzewie gwarantuje, że dla któregoś z pól dziedziny zostaną puste, to oznacza, że dodanie węzła nie przyniesie pożądanych rezultatów, gdyż w przyszłości natknie się na brak możliwości rozwiązań.

W takiej sytuacji pomija się taki węzeł i szuka innych rozwiązań.

1.3 Heurystyki

Heurystyki są kluczowym aspektem w działaniu obu algorytmów - to one definiują kolejność wyboru kolejnych pól do wypełnienia, a także kolejność wartości do sprawdzenia. Ich zmiana może znacząco wpływać na skuteczność algorytmów oraz czas ich wykonywania. Przy dobrej optymalizacji są w stanie zapewnić zadowalające wyniki w odpowiednim czasie.

1.3.1 Heurystyki wyboru pola

Zaimplementowano trzy heurystyki wyboru pola:

1. **Wybór w kolejności definicji**
   Taka heurystyka zapewnia wybór pierwszego pustego pola w Sudoku - sprawdza wierszami.

2. **Wybór pola najbardziej ograniczonego**
   Heurystyka gwarantuje wybór takiego pola, które jest najbardziej ograniczone, tzn. ma najmniejszy zbiór możliwych wartości do wpisania - najmniejszą dziedzinę.

3. **Wybór najbardziej ograniczonego pola w oknie, które ma już zdefiniowane jak najwięcej pól**
   Ta heurystyka w pierwszej kolejności wybiera okno, które ma jak najmniej pustych pól (dzięki temu łatwiej odszukać ich wartość), a następnie wybiera to pole, które jest najbardziej ograniczone, a tym samym - jego wartość będzie najbardziej oczywista.

1.3.2 Heurystyki wyboru wartości pola

Zaimplementowano trzy heurystyki wyboru wartości pola:

1 **Uporządkowany wybór**
Taka heurystyka zapewnia wybór pierwszej wartości z rosnąco uporządkowanej dziedziny danego pola.

2 **Losowy wybór**
Heurystyka gwarantuje wybór wartości z dziedziny w sposób losowy.

3 **Wybór wartości najrzadziej powtarzającej się w dziedzinach danego wiersza Sudoku**
Ta heurystyka wybiera taką wartość, która będzie najbardziej oczywista - czyli taką, która najrzadziej występuje w dziedzinach wspólnego wiersza. Gwarantuje to, że najpierw sprawdzone zostaną wartości najbardziej prawdopodobne.

2 Badania wpływu poszczególnych heurystyk na działanie algorytmów

Wszystkie badania zostaną przeprowadzone dla trzech łamigłówek o rożnych poziomach:

1. Średnie (Sudoku 23)
2. Trudne (Sudoku 36)
3. Mieszane (Sudoku 42)

2.1 Sudoku 23 - średnie

1 Czas znalezienia pierwszego rozwiązania

Badanie ma na celu porównanie czasów znalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 1: Czas znalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	1154	361	1527
	MostRestrictive	889	189	1367
	MostRestInGrid	435	442	828
Forward	InOrder	1307	69	1538
	MostRestrictive	898	84	1261
	MostRestInGrid	392	620	607

2 Czas znalezienia wszystkich rozwiązań

Badanie ma na celu porównanie czasów znalezienia wszystkich rozwiązań Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 2: Czas znalezienia wszystkich rozwiązań

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	1326	1310	1668
	MostRestrictive	1010	1000	1516
	MostRestInGrid	2162	2002	4456
Forward	InOrder	1448	1358	1770
	MostRestrictive	1014	958	1427
	MostRestInGrid	2377	2451	3175

3 Liczba powrotów do momentu odnalezienia pierwszego rozwiązania

Badanie ma na celu porównanie liczby powrotów do momentu odnalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 3: Liczba powrotów do momentu odnalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	194488	40559	194488
	MostRestrictive	194488	24097	194488
	MostRestInGrid	58488	75205	58488
Forward	InOrder	61580	1284	61580
	MostRestrictive	38580	1318	38580
	MostRestInGrid	16258	26035	16258

4 Całkowita liczba powrotów

Badanie ma na celu porównanie całkowitej liczby powrotów w Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 4: Całkowita liczba powrotów

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	222592	222592	222592
	MostRestrictive	222592	222592	222592
	MostRestInGrid	469880	469880	469880
Forward	InOrder	70286	70286	70286
	MostRestrictive	45769	45769	45769
	MostRestInGrid	127809	127809	127809

5 Liczba odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania

Badanie ma na celu porównanie liczby odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 5: Liczba odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	24338	5101	24338
	MostRestrictive	24338	3040	24338
	MostRestInGrid	7338	9428	7338
Forward	InOrder	24282	541	24282
	MostRestrictive	16397	597	16397
	MostRestInGrid	6559	10370	6559

6 Całkowita liczba odwiedzonych węzłów

Badanie ma na celu porównanie całkowitej liczby odwiedzonych węzłów w Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 6: Całkowita liczba odwiedzonych węzłów

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	27824	27824	27824
	MostRestrictive	27824	27824	27824
	MostRestInGrid	58735	58735	58735
Forward	InOrder	27768	27768	27768
	MostRestrictive	19613	19613	19613
	MostRestInGrid	49640	49640	49640

2.2 Sudoku 36 - trudne

1 Czas znalezienia pierwszego rozwiązania

Badanie ma na celu porównanie czasów znalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 7: Czas znalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	805	375	1190
	MostRestrictive	845	703	1310
	MostRestInGrid	10597	11676	11094
Forward	InOrder	648	624	1009
	MostRestrictive	2206	1633	2863
	MostRestInGrid	13008	7945	11290

2 Czas znalezienia wszystkich rozwiązań

Badanie ma na celu porównanie czasów znalezienia wszystkich rozwiązań Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 8: Czas znalezienia wszystkich rozwiązań

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	957	1013	1432
	MostRestrictive	978	1115	1613
	MostRestInGrid	15404	12259	15939
Forward	InOrder	767	805	1169
	MostRestrictive	2902	3065	3505
	MostRestInGrid	20895	16279	18880

3 Liczba powrotów do momentu odnalezienia pierwszego rozwiązania

Badanie ma na celu porównanie liczby powrotów do momentu odnalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 9: Liczba powrotów do momentu odnalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	116758	36739	116758
	MostRestrictive	116758	88851	116758
	MostRestInGrid	2039510	2801000	2039510
Forward	InOrder	41219	34708	41219
	MostRestrictive	106715	66733	106715
	MostRestInGrid	910832	782390	910832

4 Całkowita liczba powrotów

Badanie ma na celu porównanie całkowitej liczby powrotów w Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 10: Całkowita liczba powrotów

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	140864	140864	140864
	MostRestrictive	140864	140864	140864
	MostRestInGrid	2988064	2988064	2988064
Forward	InOrder	50141	50141	50141
	MostRestrictive	130476	130476	130476
	MostRestInGrid	1503007	1503007	1503007

5 Liczba odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania

Badanie ma na celu porównanie liczby odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 11: Liczba odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	14622	4619	14622
	MostRestrictive	14622	11137	14622
	MostRestInGrid	254966	350154	254966
Forward	InOrder	14483	12128	14483
	MostRestrictive	55506	34942	55506
	MostRestInGrid	343818	301371	343818

6 Całkowita liczba odwiedzonych węzłów

Badanie ma na celu porównanie całkowitej liczby odwiedzonych węzłów w Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 12: Całkowita liczba odwiedzonych węzłów

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	17608	17608	17608
	MostRestrictive	17608	17608	17608
	MostRestInGrid	373508	373508	373508
Forward	InOrder	17460	17460	17460
	MostRestrictive	67957	67957	67957
	MostRestInGrid	548704	548704	548704

2.3 Sudoku 42 - mieszane

1 Czas znalezienia pierwszego rozwiązania

Badanie ma na celu porównanie czasów znalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 13: Czas znalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	283	181	634
	MostRestrictive	399	272	676
	MostRestInGrid	-	-	-
Forward	InOrder	248	356	339
	MostRestrictive	1137	3694	1337
	MostRestInGrid	-	-	-

2 Czas znalezienia wszystkich rozwiązań

Badanie ma na celu porównanie czasów znalezienia wszystkich rozwiązań Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 14: Czas znalezienia wszystkich rozwiązań

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	2828	4099	6594
	MostRestrictive	4324	4419	7009
	MostRestInGrid	-	-	-
Forward	InOrder	2593	2720	2942
	MostRestrictive	11048	10504	12228
	MostRestInGrid	-	-	-

3 Liczba powrotów do momentu odnalezienia pierwszego rozwiązania

Badanie ma na celu porównanie liczby powrotów do momentu odnalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 15: Liczba powrotów do momentu odnalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	44087	17251	44087
	MostRestrictive	44087	25166	44087
	MostRestInGrid	-	-	-
Forward	InOrder	8734	12243	8734
	MostRestrictive	49653	212871	49653
	MostRestInGrid	-	-	-

4 Całkowita liczba powrotów

Badanie ma na celu porównanie całkowitej liczby powrotów w Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 16: Całkowita liczba powrotów

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	777841	777841	777841
	MostRestrictive	777841	777841	777841
	MostRestInGrid	-	-	-
Forward	InOrder	153769	153769	153769
	MostRestrictive	644129	644129	644129
	MostRestInGrid	-	-	-

5 Liczba odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania

Badanie ma na celu porównanie liczby odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 17: Liczba odwiedzonych węzłów do momentu znalezienia pierwszego rozwiązania

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	5542	2188	5542
	MostRestrictive	5542	3177	5542
	MostRestInGrid	-	-	-
Forward	InOrder	5259	6493	5259
	MostRestrictive	31525	126512	31525
	MostRestInGrid	-	-	-

6 Całkowita liczba odwiedzonych węzłów

Badanie ma na celu porównanie całkowitej liczby odwiedzonych węzłów w Sudoku. Poniżej zestawiono uśrednione na podstawie 10 uruchomień wyniki:

Tabela 18: Całkowita liczba odwiedzonych węzłów

Algorithm	CellSelection	InOrder ValueSelection	RandomCell ValueSelection	LeastInRowCell ValueSelection
Backtrack	InOrder	97235	97235	97235
	MostRestrictive	97235	97235	97235
	MostRestInGrid	-	-	-
Forward	InOrder	93369	93369	93369
	MostRestrictive	406091	406091	406091
	MostRestInGrid	-	-	-

3 Analiza

1 Czas znalezienia pierwszego rozwiązania

Dla łamigłówek średnich zauważa się znaczną poprawę w czasach znajdowania pierwszego rozwiązania wraz ze wzrostem skomplikowania heurystyk wyboru pól do wypełnienia, jednak w kombinacji z heurystyką losowego wyboru wartości pola *RandomCellValueSelection* czas ten się wydłuża.

Jest to spowodowane tym, że używając heurystyk wyboru pola, które im bardziej skomplikowane, tym docelowo lepiej dobierają pola i jednoczesnym wprowadzaniu losowości w momencie wyboru wartości, powoduje się pewien konflikt - wyestymowany wcześniej odpowiedni wybór pola jest burzony poprzez losowy (i tym samym nienajlepszy) dobór wartości, co wiąże się z niepotrzebnym nakładem obliczeniowym, niż w przypadku, gdyby ten dobór był deterministyczny (lepszy).

Dbając o jak najlepszy czas działania algorytmów, powinno się dobrać te dwie heurystyki w taki sposób, aby one szły ze sobą w parze, tzn. jedna powinna dopełniać idei drugiej, zamiast prezentować inne podejście i "rozsynchronizowywać" wyniki.

Wyżej wymieniona zależność jest widoczna dla obu algorytmów, ale tylko dla łamigłówek średnich - dla trudnych i mieszanych czas zdecydowanie wzrasta wraz ze wzrostem stopnia skomplikowania heurystyki wyboru pola. Jest to spowodowane tym, że metody restrykcyjne (w tym szczególnie restrykcyjna dla okna Sudoku) wiążą się z dużym nakładem pracy w porównaniu do np. *InOrderCellSelection*. I choć parametry lepiej są wybierane, tak więcej czasu zabiera ich wybór. Dodatkowo, Sudoku mieszane ma znacznie więcej niż 1 rozwiązanie, stąd też obserwuje się duży nakład czasowy.

Zależność ta jest widoczna zarówno dla algorytmu *BacktrackAlgorithm*, jak i dla *ForwardCheckingAlgorithm*.

2 Czas znalezienia wszystkich rozwiązań

Podobnie jak w powyższym, całkowity czas znalezienia rozwiązań wiąże się w różnym nakładem pracy w zależności od wybranych heurystyk. Te najmniej skomplikowane okazują się najszybsze (dla wszystkich algorytmów i łamigłówek).

3 Liczba powrotów do momentu odnalezienia pierwszego rozwiązania

Jak już wspomniano wyżej, czas poświęcany na obliczenia skutkuje lepszymi wynikami w postaci liczby powrotów. Dla algorytmów średnich występuje mniej nawrotów do momentu odnalezienia pierwszego rozwiązania. Oznacza to, że algorytm poprzez odpowiednią ekstrakcje i wybór wartości przybywa krótszą drogę w znalezieniu rozwiązania.

Zależność ta zanika jednak dla trudnych i mieszanych łamigłówek rośnie liczba nawrotów ze względu na skomplikowanie zadań - rozwiązania nie są tak oczywiste jak w łatwiejszych łamigłówkach i rzadko występuje okazja do bardzo wczesnego nawrotu (i tym samym skrócenia obliczeń). Trzeba stosunkowo dużo wartości przejrzeć ze względu na większą liczbę pustych pól w zadanym Sudoku.

Dla najtrudniejszych Sudoku zrezygnowano ze sposobu ich rozwiązania z użyciem najbardziej skomplikowanej i obciążającej heurystyki wyboru pola ze względu na zbyt długi czas wykony-

wania obliczeń.

4 Całkowita liczba powrotów
Całkowita liczba powrotów również rośnie wraz ze wzrostem skomplikowania heurystyki wyboru pola Sudoku, najbardziej znacząco dla algorytmów *ForwardCheckingAlgoritm*. Algorytm *BacktrackAlgorithm* gwarantuje bardzo porównywalną liczbę nawrotów dla heurystyk *InOrderCellSelection* oraz *MostRestrictiveCellSelection*, ale wzrasta dla najbardziej skomplikowanej heurystyki - *MostRestrictiveInGridCellSelection*.

4 Porównanie algorytmów Backtrack i ForwardChecking

Do porównania użyto względnie najlepszych kombinacji heurystyk (dla sprawdzanych Sudoku łatwych - numer 7):

1. InOrderValueSelection
2. MostRestrictiveInGridCellSelection

Tabela 19: Porównanie algorytmów dla Sudoku łatwego

Algorithm	Time to 1st	Time to all	Backtracks to 1st	All backtracks	Nodes to 1st	All nodes
Backtrack	554	1007	109002	212584	13655	26573
Forward	705	1202	37289	70747	16923	32287

Do porównania użyto względnie najlepszych kombinacji heurystyk (dla sprawdzanych Sudoku mieszanych2 - numer 44):

1. LeastOccurencedInRowCellValueSelection
2. MostRestrictiveCellSelection

Tabela 20: Porównanie algorytmów dla Sudoku łatwego

Algorithm	Time to 1st	Time to all	Backtracks to 1st	All backtracks	Nodes to 1st	All nodes
Backtrack	0	10338	0	1869305	0	233663
Forward	0	13676	0	913094	0	440103

5 Podsumowanie

Chcąc osiągnąć jak najlepsze rezultaty należy odpowiednio zadbać o dobór heurystyk. Niektóre dobrze ze sobą współpracują, a niekiedy stosowanie niektórych razem nie ma sensu ze względu na inne ich przeznaczenie.

Ważną kwestią jest także wybór samego algorytmu rozwiązującego łamigłówkę. *ForwardChecking* jest lepszy - lepiej dobiera parametru poprzez fakt, że ma możliwość przerwania pracy na danym poddrzewie wcześniej niż *Backtrack*, który pracę przerywa dopiero w momencie, kiedy nie ma możliwości takiego doboru parametru, żeby wszystkie ograniczenia dla sudoku były spełnione. Dzieje się to dopiero w trakcie próby dodania aktualnego węzła do poddrzewa.

ForwardChecking takie szacunki wykonuje krok wcześniej, dzięki czemu na wcześniejszym etapie ma możliwość odrzucenia poddrzew, które na pewno nie spełnią oczekiwanego rezultatu w postaci rozwiązania sudoku.

Stąd też, lepiej jest stosować algorytm *ForwardChecking*, jednak *Backtrack* jest także sensownym wyborem dla prostych problemów, do których nie są wymagane niepotrzebnie skomplikowane obliczenia. Przy okazji, wybierając *Backtrack* w takim przypadku najpewniej będzie można osiągnąć szybszy czas rozwiązania łamigłówki, właśnie ze względu na jego prostotę w działaniu.