JCSI 2 (2016) 140-143
Journal of Computer Sciences Institute

Wysłane: 2016-11-29
Przyjęte: 2016-11-30

Analiza porównawcza algorytmów rozwiązujących Sudoku

Emil Wnuk*, Edyta Łukasik
Instytut Informatyki, Politechnika Lubelska, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia analizę porównawczą wybranych algorytmów, służących do rozwiązywania Sudoku. Porównywane są puzzle różnej wielkości, posiadające różny poziom trudności. Sprawdzany jest wpływ tych czynników na czas działania algorytmów. Na potrzeby badań stworzono specjalną aplikację, która zawiera wszystkie potrzebne algorytmy.

Słowa kluczowe: Sudoku; algorytmy; czas działania

*Autor do korespondencji.

Adres e-mail: emil.wnuk@gmail.com

Comparative analysis of algorithms for solving Sudoku

Emil Wnuk*, Edyta Łukasik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents a comparative analysis of selected algorithms for solving Sudoku. Puzzles of different sizes and having different levels of difficulty are compared. The impact of these factors on the duration of the algorithms is examined. For the study, a special application that contains all the necessary algorithms has been created.

Keywords: Sudoku; algorithms; solving time

*Corresponding author.

E-mail address: emil.wnuk@gmail.com

1. Wstęp

Wbrew powszechnej opinii, Sudoku nie zostało wynalezione w Japonii. Pierwsza publikacja łamigłówki miała miejsce w Stanach Zjednoczonych w 1979 roku [1]. Do kraju samurajów puzzle dotarły siedem lat później, jednak międzynarodową popularność zyskały dopiero w roku 2005 [2]. Nazwa układanki pochodzi od japońskiego wyrażenia oznaczającego: „cyfry muszą być pojedyncze” [3].

Klasyczna wersja Sudoku ma kształt diagramu 9 × 9, który dzieli się na dziewięć kwadratów 3 × 3 [4]. Celem łamigłówki jest wypełnienie schematu tak, aby w każdym wierszu, każdej kolumnie oraz każdym kwadracie 3 × 3 znalazło się dokładnie po jednej cyfrze z zakresu od 1 do 9 [5]. Cyfry mają tu znaczenie umowne, ponieważ inne symbole (litery, znaki, kolory) również mogą być stosowane [6].

Każde Sudoku zaprojektowane jest tak, aby posiadało dokładnie jedno rozwiązanie [7]. Podczas wyszukiwania brakujących cyfr nie można zmieniać już istniejących w danym diagramie. Aby poprawnie rozwiązać Sudoku trzeba dokładnie, krok po kroku, wykonywać pewne żmudne algorytmy. Pośpiech i brak umiejętności logicznego myślenia sprzyjają popełnianiu błędów, które często zauważane są dopiero w końcowym etapie rozwiązywania. Taka sytuacja sprzyja frustracji uczestnika gry, dlatego bardzo pomocny okazuje się komputer – idealne narzędzie do rozwiązywania tego typu zagadek.

W przeprowadzonych testach analizowano czas rozwiązywania Sudoku za pomocą algorytmów komputerowych. Zbadano puzzle różnej wielkości:

- *s2* (4 × 4);
- *s3* (9 × 9);
- *s4* (16 × 16);
- *s5* (25 × 25);

posiadające różny poziom trudności:

- *low* (niski);
- *med* (średni);
- *top* (wysoki);
- *xxx* (ekstremalny).

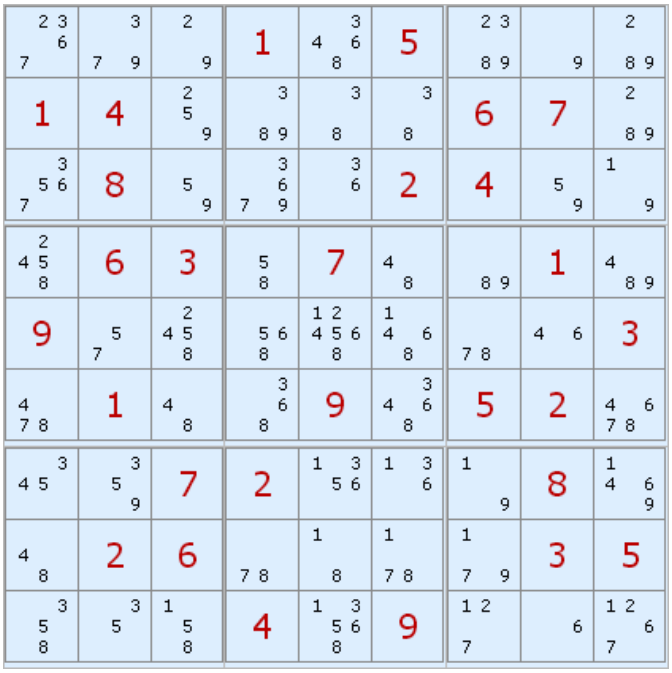
Na potrzeby badań stworzono specjalną aplikację, która zawiera wszystkie potrzebne algorytmy:

1) Brute Force (BF)
2) Brute Force Fast (BFF)
3) Backtracking (BT)
4) Backtracking Fast (BTF)
5) Naked Single (NS)
6) Naked Single Backtracking (NSBT)
7) Hidden Single (HS)
8) Hidden Single Backtracking (HSBT)
9) Multiple Choice (MC)

W przypadku, gdy czas działania algorytmu przekroczył 60 sekund, przerywano jego wykonywanie, a do wyniku przypisywano symbol *inf*. W przypadku, gdy rozwiązanie okazało się niepoprawne, wynikiem stawał się symbol *err*.

2. Brute Force

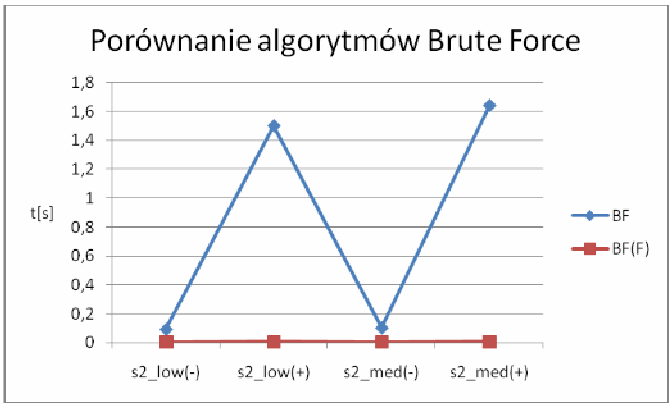
Algorytm Brute Force (BF) to prosty algorytm naiwny, który wpisuje w puste pola wszystkie kombinacje cyfr od 1 do 9 i za każdym razem sprawdza, czy rozwiązanie jest poprawne. W przykładowym Sudoku (rysunek 1) znajduje się 28 cyfr początkowych. Do wstawienia pozostało 53 cyfry, zatem algorytm musi sprawdzić aż 9^{53} przypadków. BF jest najwolniejszym z zaimplementowanych algorytmów. Jego zaletami są prostota i pewność, że wszystkie prawidłowe rozwiązania zostaną znalezione [8].



Rys. 1. Przykładowe Sudoku

Brute Force Fast (BFF) jest ulepszoną wersją algorytmu BF, która jest dużo szybsza dzięki temu, że w puste pola wstawiane są tylko te wartości, które nie kolidują z cyframi początkowymi. Wykaz tych wartości przedstawia Rysunek 1.

Na rysunku 2 przedstawiono wykres z czasami działania algorytmów BF i BFF. Testy przeprowadzono na Sudoku drugiego stopnia (4 × 4).



Rys. 2. Porównanie algorytmów Brute Force

3. Backtracking

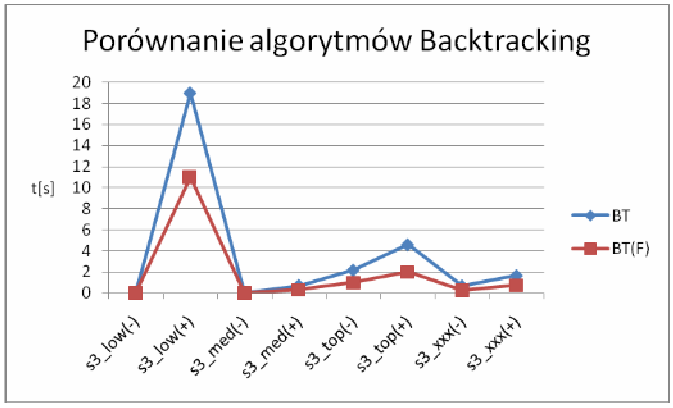
Algorytm Backtracking (BT), podobnie jak BF, także korzysta ze wszystkich cyfr od 1 do 9, ale po każdym wstawieniu cyfry sprawdza, czy nie koliduje ona z pozostałymi wartościami znajdującymi się na planszy [9].

Jeśli wystąpi sprzeczność, to dana wartość jest usuwana, a na jej miejsce zostaje wstawiona kolejna. Pokazane jest to na rysunku 3, gdzie cyfra znajdująca się w polu (8,2) musi zostać zastąpiona, ponieważ koliduje z polem (2,2) oraz (7,7).

Rys. 3. Działanie algorytmu Backtracking

Backtracking Fast (BTF) jest ulepszoną wersją algorytmu BT, która (podobnie jak algorytm BFF) korzysta z wartości, które nie kolidują z cyframi początkowymi. Dzięki temu jest najszybszym spośród zaimplementowanych algorytmów naiwnych (BF, BFF, BT, BTF). Złożoność obliczeniowa algorytmów naiwnych jest postaci potęgowej.

Rysunek 4 przedstawia porównanie algorytmów BT i BTT, przeprowadzone na Sudoku trzeciego stopnia (9 × 9).

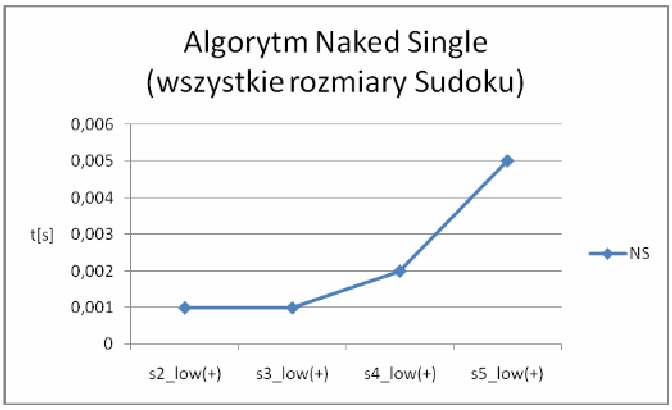


Rys. 4. Porównanie algorytmów Backtracking

4. Algorytmy eliminacyjne

Naked Single (NS) to algorytm, który stosuje metodę eliminacji. Jeżeli w danym polu znajduje się tylko jeden kandydat, wartość ta zostaje wstawiona na stałe. W przykładzie na rysunku 1 takimi polami są (8,1) i (8,9). Nowo powstałe wartości wykorzystywane są do eliminacji następnych. Jeżeli przynajmniej jedna cyfra zostanie wstawiona, to algorytm wykonuje się jeszcze raz, w przeciwnym wypadku działanie zostaje przerwane [10].

Na rysunku 5 znajduje się wykres przedstawiający wyniki algorytmu NS ze wszystkich rozmiarów Sudoku.

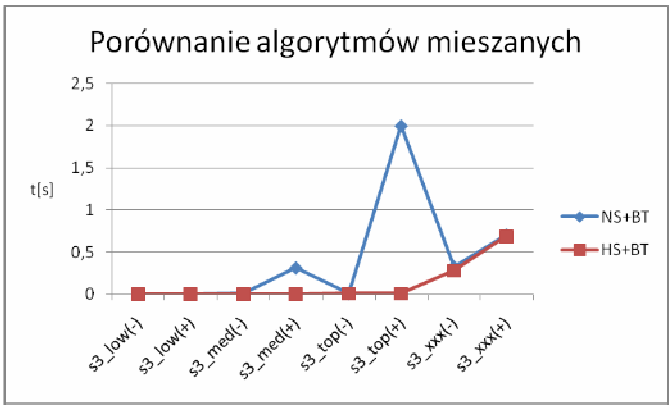


Rys. 5. Algorytm Naked Single

Algorytm Hidden Single (HS) jest zbliżony do NS, który wyszukiwał komórkę z jednym kandydatem. HS przeszukuje wiersze, kolumny i kwadraty w celu znalezienia takiego ciągu, gdzie dany kandydat występuje tylko raz [11].

5. Algorytmy mieszane

Na rysunku 6 znajdują się wykresy algorytmów Naked Single Backtracking (NSBT) oraz Hidden Single Backtracking (HSBT). NSBT jest połączeniem dwóch algorytmów: NS i BT z tym, że BT uruchamiany jest na wartościach, które nie zostały wyeliminowane, lecz tylko i wyłącznie wtedy, gdy NS nie uzyskał rozwiązania. W przypadku HSBT sposób działania jest analogiczny do NSBT. Najpierw uruchomiony jest HS, a następnie BT.

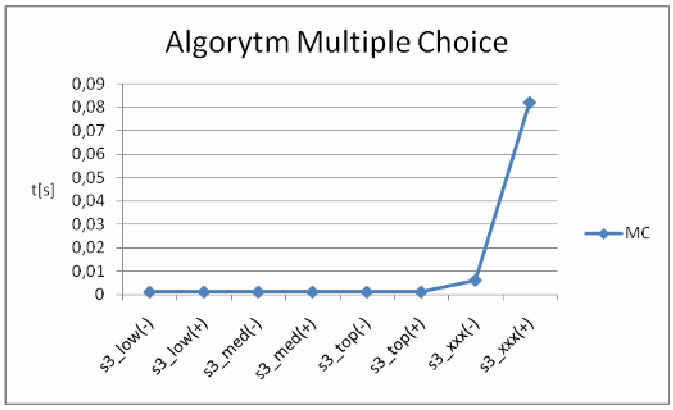


Rys. 6. Porównanie algorytmów mieszanych

Algorytmy mieszane to połączenie algorytmów eliminacyjnych (NS i HS) z naiwnym BT. Na wykresie (rysunek 6) widać, że na łatwych poziomach trudności czas wykonania jest zbliżony do 0. Na trudniejszych poziomach występuje tendencja rosnąca z tym, że HSBT ma bardziej przewidywalny przebieg w porównaniu do NSBT.

Multiple Choice (MC) to połączenie trzech algorytmów: NS, HS oraz specjalnego typu Backtrackingu, który polega na tym, że po wykonaniu NS i HS, algorytm wybiera pole posiadające dwóch kandydatów i dzieli się na dwie rekurencyjne części. Każda część uznaje swojego kandydata za prawidłowego i działa w oparciu na tej tezie. Po wykryciu sprzeczności następuje powrót to wyższego poziomu rekursji.

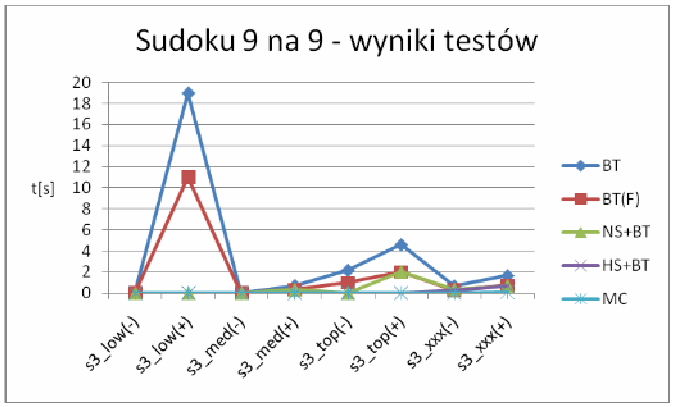
Na rysunku 7 znajduje się wykres przedstawiający wyniki algorytmu MC.



Rys. 7. Algorytm Multiple Choice

Sudoku na wykresie na rys. 7 są uporządkowane w kolejności od najłatwiejszego do najtrudniejszego i w tym wypadku czasy działania algorytmów potwierdziły tę tezę.

Podsumowanie testów dla Sudoku trzeciego stopnia przedstawia wykres na rysunku 8.



Rys. 8. Sudoku 9 na 9 – wyniki testów

Zbiorcze zestawienie wszystkich wyników przedstawia tabela 1.

Tabela 1. Zbiorcze zestawienie wszystkich wyników

Sudoku	BF	BF(F)	BT	BT(F)	NS	NS+BT	HS	HS+BT	MC
s2_low(-)	0,094	0,004	0,005	0,004	0,001	0,001	0,001	0,001	0,001
s2_low(+)	1,500	0,006	0,005	0,004	0,001	0,001	0,001	0,001	0,001
s2_med(-)	0,104	0,004	0,004	0,004	0,001	0,001	0,001	0,001	0,001
s2_med(+)	1,640	0,006	0,004	0,004	0,001	0,001	0,001	0,001	0,001
s3_low(-)	inf	inf	0,008	0,008	0,001	0,001	0,001	0,001	0,001
s3_low(+)	inf	inf	19,00	11,00	0,001	0,001	0,001	0,001	0,001
s3_med(-)	inf	inf	0,014	0,008	err	0,008	0,001	0,001	0,001
s3_med(+)	inf	inf	0,719	0,315	err	0,314	0,001	0,001	0,001
s3_top(-)	inf	inf	2,172	0,984	err	0,007	err	0,007	0,001
s3_top(+)	inf	inf	4,600	1,984	err	2,000	err	0,007	0,001
s3_xxx(-)	inf	inf	0,672	0,297	err	0,328	err	0,281	0,006
s3_xxx(+)	inf	inf	1,656	0,734	err	0,703	err	0,688	0,082
s4_low(-)	inf	inf	inf	inf	0,002	0,002	0,002	0,002	0,002
s4_low(+)	inf	inf	inf	inf	0,002	0,002	0,002	0,002	0,002
s4_med(-)	inf	inf	inf	inf	err	0,015	0,002	0,002	0,002
s4_med(+)	inf	inf	inf	inf	err	inf	err	inf	0,828
s4_top(-)	inf	inf	inf	inf	err	inf	err	0,254	0,010
s4_top(+)	inf	inf	inf	inf	err	inf	err	inf	inf
s5_low(-)	inf	inf	inf	inf	0,004	0,004	0,004	0,004	0,004
s5_low(+)	inf	inf	inf	inf	0,005	0,005	0,005	0,005	0,005
s5_med(-)	inf	inf	inf	inf	err	inf	err	inf	inf
s5_med(+)	inf	inf	inf	inf	err	inf	err	inf	inf
s5_top(-)	inf	inf	inf	inf	err	inf	err	inf	inf
s5_top(+)	inf	inf	inf	inf	err	inf	err	inf	inf
inf - czas działania algorytmu przekroczył 60 sekund									
err - algorytm zakończył działanie (wynik niepoprawny)									

6. Wnioski

Algorytmy Brute Force (BF i BFF) uzyskały czasy poniżej dwóch sekund dla wszystkich Sudoku drugiego stopnia.

Algorytmy Backtracking (BT i BTF) uzyskały czasy poniżej dwudziestu sekund dla wszystkich Sudoku 9 × 9.

Algorytmy eliminacyjne (NS i HS) wykonały się poprawnie dla wszystkich Sudoku z poziomu *low*. Były to również jedyne algorytmy, które zwróciły zły wynik.

Algorytm MC uzyskał najlepszy wynik w każdym z rozpatrywanych przykładów – oznacza to, że jest najbardziej optymalnym algorytmem. Jest to również jedyny algorytm, który dał poprawne rozwiązanie w przykładzie *s4_med(+)*.

Hierarchia algorytmów pod względem czasów wykonania przedstawia się następująco: BF > BT > BFF > BTF > NS > NSBT > HS > HSBT > MC.

Literatura

[1] Wilson R.J., Jak rozwiązywać sudoku, Dom Wydawniczy Rebis, Poznań 2005

[2] H. Intelm, How to Solve Every Sudoku Puzzle, Vol.2, Geostar Publishing LLC. 2005.

[3] N. Jussien, A-Z of Sudoku, ISTE Ltd., USA, 2007.

[4] W.-M. Lee, Programming Sudoku, Apress, USA, 2006.

[5] Arnoldy, Ben. "Sudoku Strategies". *The Home Forum*. The Christian Science Monitor.

[6] Gordon Royle, The University of Western Australia. Minimum Sudoku.

[7] M.H.Alsuwaiyel, Algorithms Design Techniques and Analysis. London: World Scientific Publishing Company. 2008.

[8] Daniel J. Bernstein, Understanding brute force. 2007.

[9] Peter van Beek, Backtracking Search Algorithms, 2006 Elsevier

[10] George T. Heineman, Algorithms in a nutshell, Apress, 2008

[11] H. Intelm, How to Solve Every Sudoku Puzzle, Vol.4, Geostar Publishing LLC. 2007.