Laboratorium 2
Problemy spełniania ograniczeń

1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z problemami spełniania ograniczeń – ich formułowaniem oraz metodami rozwiązywania - na przykładzie Sudoku i Jolki.

2. Wstęp

Problem CSP (Constrained Satisfaction Problem) – jest to z zagadnienie przeszukiwania z ograniczeniami, albo z więzami. Stanowi ono specyficzną grupę problemów przeszukiwania w przestrzeni stanów, w której zdefiniowany jest :

- skończony zbiór zmiennych,
- dla każdej zmiennej zdefiniowany jest skończony zbiór możliwych jej wartości, zwany dziedziną,
- skończony zbiór ograniczeń (constraints), zwanych również więzami.

Rozwiązaniem problemu CSP jest każda kombinacja wartości zmiennych, która spełnia wszystkie więzy.

Algorytm z nawrotami – jest to ogólny algorytm wyszukiwania wszystkich lub kilku rozwiązań niektórych problemów obliczeniowych, który stopniowo generuje kandydatów na rozwiązanie. Jeśli algorytm stwierdzi, że znaleziony kandydat nie może być poprawnym rozwiązaniem, nawraca do punktu, gdzie może podjąć inną decyzje związaną z jego budową.

Metoda sprawdzania w przód – jest bardzo podobna do algorytmu z nawrotami jednak dodatkowo monitoruje i kontroluje dziedzinę problemu dla każdego elementu ze zbioru. Dzięki temu metoda ta pozwala na wcześniejsze wykrywanie błędnych kroków.

3. Opis projektu i analiza wyników

W ramach projektu zrealizowano następujące zadania:

- sformułowano łamigłówkę Sudoku oraz Jolka jako CSP wraz z implementacją i wczytaniem danych,
- zaimplementowano algorytm metody przeszukiwania z nawrotami,
- zaimplementowano jedną heurystykę wyboru zmiennej i jedną heurystyki wyboru wartości,
- zaimplementowano metodę sprawdzania w przód,
- zbadano działania implementacji dla obu łamigłówek,
- porównano działanie obu metod rozwiązywania dla Sudoku i Jolki.

Do implementacji wykorzystano język C++. **Przy badaniu wyników dla danej metody brano pod uwagę 10 uruchomień algorytmu z tymi samymi parametrami.**
Algorytm przeszukiwania z nawrotami jest bardzo dobrym narzędziem do rozwiązywania problemów CSP. W ramach ćwiczenia zrealizowana wersja opierała się na heurystyce wyboru zmiennych oraz wartości w pozycji kolejności definicji. Wyniki testowania programu służącego do rozwiązywania Sudoku zamieszczono w tabeli 1. Testów dokonano na:

- 1 pliku z grupy problemów łatwych (id = 6),
- 1 pliku z grupy problemów średnich (id = 22),
- 1 pliku z grupy problemów trudnych (id = 37),
- 1 pliku z grupy problemów mieszanych1 (id = 43),
- 1 pliku z grupy problemów mieszanych2 (id = 44).

Analogiczne czynności wykonano dla algorytmu sprawdzania w przód (tabela 2).

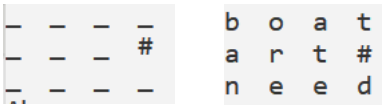
ID zbioru	Poziom trudności	Czas znalezienia pierwszego rozwiązania [s]	Liczba odwiedzonych węzłów do znalezienia pierwszego rozwiązania	Liczba nawrotów do znalezienia pierwszego rozwiązania	Liczba rozwiązań	Całkowita liczba odwiedzonych węzłów drzewa	Całkowita liczba nawrotów	Człkowity czas metody [s]
6	1	0.0011187	7925	7596	1	7925	7596	0.0011187
22	4	0.0029416	30562	28759	1	30562	28759	0.0029416
37	7	0.0118929	107431	103376	1	107431	103376	0.0118929
43	8	0.0154052	149293	140369	87	1273308	1198479	0.42012
44	9	-	-	-	BRAK	-	-	-

Tabela 1. Wyniki przeszukiwania z nawrotami dla Sudoku

Jak można zauważyć, wyniki działania algorytmów znacznie się od siebie różnią. Dla zbiorów o względnie niewielkim stopniu skomplikowania, różnice te nie są bardzo znaczące, jednak dla problemów trudnych, są bardzo istotne. Poprawne rozwiązania zostały znalezione dla wszystkich zadań z wyjątkiem zbioru o ID = 44. Łamigłówka ta nie posiadała rozwiązania, ponieważ nie spełniała ona podstawowych kryteriów Sudoku. Zbiory o ID 6, 22, 37 posiadały jedno rozwiązanie, zaś zbiór o ID = 43 posiadał ich 87. Wraz ze wzrostem stopnia skomplikowania czas potrzebny na rozwiązanie zadania ulegał wydłużeniu, jednak rozwiązanie nawet najtrudniejszych problemów zajmowało mniej niż 1 sekundę. Na rysunku 1, 2, 3 przedstawiono przykładowe zbiory testowe wraz z wygenerowanymi rozwiązaniami.

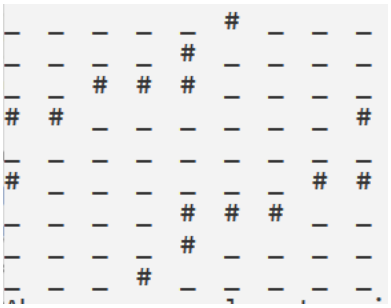
ID zbioru	Poziom trudności	Czas znalezienia pierwszego rozwiązania [s]	Liczba odwiedzonych węzłów do znalezienia pierwszego rozwiązania	Liczba nawrotów do znalezienia pierwszego rozwiązania	Liczba rozwiązań	Całkowita liczba odwiedzonych węzłów drzewa	Całkowita liczba nawrotów	Człkowity czas metody [s]
6	1	0.0007969	1548	1255	1	1548	1255	0.0007969
22	4	0.0027349	6383	4580	1	6383	4580	0.0027349
37	7	0.0109153	21763	17708	1	21763	17708	0.0109153
43	8	0.0154751	32206	24282	87	1198479	288884	0.206638
44	9	-	-	-	BRAK	-	-	-

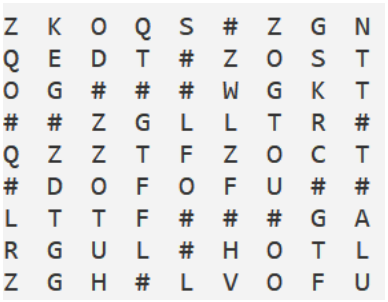
Tabela 2. Wyniki metody sprawdzania w przód dla Sudoku



Rys 1. Zbiór Jolka “puzzle0” przed i po uzupełnieniu

Rys 2. Zbiór Jolka “puzle1” przed i po uzupełnieniu





Rys 3. Zbiór Jolka “puzzle2” przed i po uzupełnieniu dla pierwszego znalezionego rozwiązania

Rozwiązanie krzyżówki Jolka było zadaniem o wiele bardziej skomplikowanym (zarówno pod względem logicznym, jak i obliczeniowym). W tabeli 3 zestawiono wyniki działania przeszukiwania z nawrotami, zaś w tabeli 4, przeszukiwania w przód. Dla najmniejszego zbioru algorytmy te działały praktycznie tak samo, jednak w miarę zwiększania się problemu różnice między rozwiązaniami stawały się bardzo znaczące.

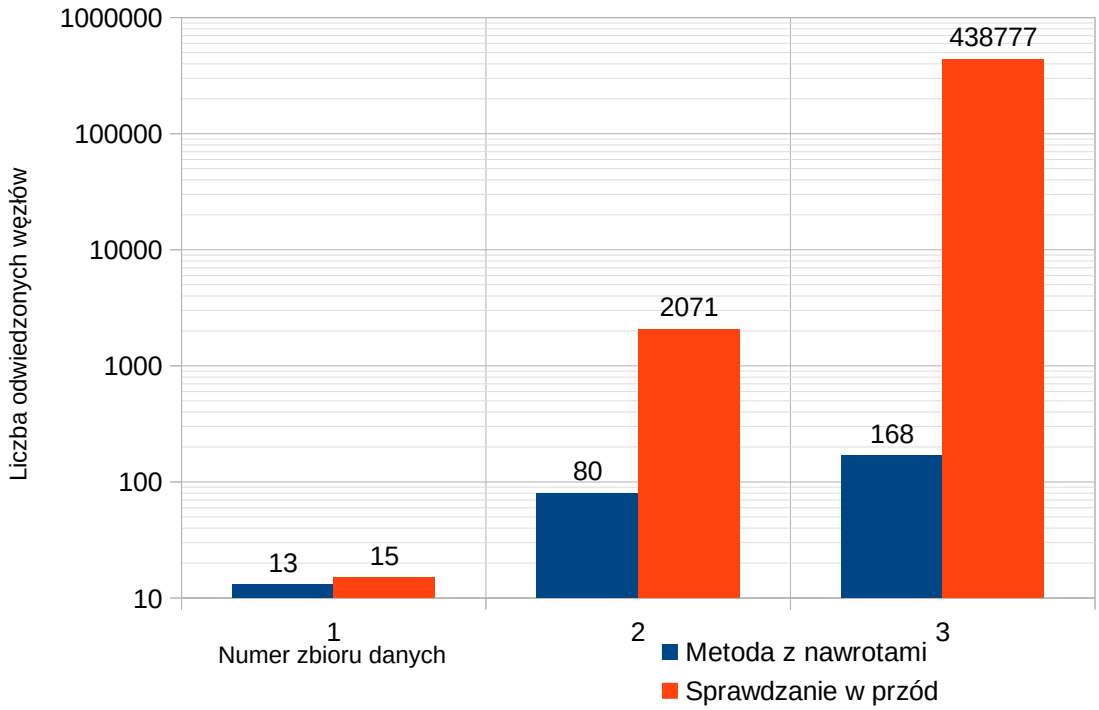
Nazwa zbioru	Czas znalezienia pierwszego rozwiązania [s]	Liczba odwiedzonych węzłów do znalezienia pierwszego rozwiązania	Liczba nawrotów do znalezienia pierwszego rozwiązania	Liczba rozwiązań	Całkowita liczba odwiedzonych węzłów drzewa	Całkowita liczba nawrotów	Człkowity czas metody [s]
puzzle0	0.0005476	15	1	1	15	1	0.0005476
puzzle1	0.0214757	2071	726	6	6296	2182	0.0588248
puzzle2	24.416	438777	410207	1	616177	497411	29.9813
puzzle3	więcej niż 1000	-	-	NIE ZNALEZIONO	-	-	więcej niż 1000
puzzle4	więcej niż 1000	-	-	NIE ZNALEZIONO	-	-	więcej niż 1000

Tabela 3. Wyniki przeszukiwania z nawrotami dla Jolki.

Dla algorytmu przeszukiwania z nawrotami nie udało się znaleźć rozwiązania dla zbioru „puzle3” oraz „puzle4” ze względu na zbyt dużą złożoność obliczeniową. Po 1000 sekund obliczenia zostały ręcznie zakończone. Kontrola dziedziny dla rozwiązywania Jolki jest sprawą kluczową. Problemy „nierozwiązywalne” dla algorytmu przeszukiwania z nawrotami są całkowicie dostępne dla metody sprawdzania w przód. Co prawda metoda ta, także nie poradziła sobie ze zbiorem „puzzle3”, jednak podczas przerywania zadania po 1000 s plansza była uzupełniona prawie całkowicie. Zestaw ten jest bardzo trudny w rozwiązaniu ze względu na występującą w nim dużą liczbę słów o tej samej długości. Na wykresie 1 przedstawiono różnice liczby odwiedzonych węzłów potrzebnych do znalezienia pierwszego rozwiązania dla obu metod na podstawie zbiorów „puzzle0”, „puzzle1”, „puzzle2”. Należy zauważyć, że na wykresie zastosowano skalę logarytmiczną.

Nazwa zbioru	Czas znalezienia pierwszego rozwiązania [s]	Liczba odwiedzonych węzłów do znalezienia pierwszego rozwiązania	Liczba nawrotów do znalezienia pierwszego rozwiązania	Liczba rozwiązań	Całkowita liczba odwiedzonych węzłów drzewa	Całkowita liczba nawrotów	Człkowity czas metody [s]
puzzle0	0.0003371	13	1	1	13	1	0.0003371
puzzle1	0.0020919	80	30	6	296	179	0.0085986
puzzle2	0.0068851	168	73	21	3273	1686	0.124122
puzzle3	więcej niż 1000	-	-	NIE ZNALEZIONO	-	-	więcej niż 1000
puzzle4	0.123358	654	65	16	11910	3058	1.82828

Tabela 4. Wyniki metody sprawdzania w przód dla Jolki



Wykres 1. Porównanie liczby odwiedzonych węzłów do znalezienia pierwszego rozwiązania dla metody z nawrotami i sprawdzania w przód. Na wykresie zestawiono wyniki z trzech przykładowych Jolek („puzzle0”, „puzzle1”, „puzzle2”)

4. Podsumowanie i wnioski

Zarówno algorytm przeszukiwania z nawrotami, jak i metoda sprawdzania w przód są dobrymi metodami rozwiązywania problemów spełniania ograniczeń, jednak ich wydajność znacznie się od siebie różni. Gdy mamy do czynienia z problemami o względnie niewielkiej złożoności, metoda poszukiwania z nawrotami może zapewniać dobre rezultaty. W przypadku niektórych zbiorów metoda ta może okazać się mało wydajna. Gdy w zbiorze znajduje się duża ilość elementów o podobnej charakterystyce względem ograniczeń, algorytm ten jest całkowicie nieskuteczny. Problemy pojawiają się także dla większych zbiorów danych. W takich przypadkach o wiele lepiej sprawdza się metoda sprawdzania w przód. Dzięki stałemu kontrolowaniu dziedziny problemu, jest ona zdolna do wcześniejszego wykrywania błędów.