Sztuczna Inteligencja i Inżynieria wiedzy

Sprawozdanie 2

Konrad Ganczuk 192151

1. Sformułowanie wybranych problemów jako problemów CSP.

Do realizacji ćwiczenia wybrałem problemy n-hetmanów, kwadratu łacińskiego oraz Sudoku.

Wszystkie z nich to problemy CSP, ponieważ muszą spełniać pewne ograniczenia.

1. Problem n-hetmanów:

Zmienne – kolumny na szachownicy {0..N}

Wartości – numer rzędu w szachownicy {0..N}

Ograniczenie – hetmani muszą być ułożeni tak, aby nie bić się nawzajem

1. Kwadrat łaciński:

Zmienne – pola w tablicy {(0, 0)…(N, N)}

Wartości – liczby {0..N}

Ograniczenie – w jednym rzędzie ani kolumnie nie może być 2 razy taka sama liczba

1. Sudoku:

Zmienne – pola w tablicy {(0, 0)…(N, N)}

Wartości – liczby {0..N}

Ograniczenia – zgodnie z zasadami gry Sudoku ta sama liczba nie może pojawić się w tym samym rzędzie, kolumnie ani kwadracie

1. Implementacja algorytmów sprawdzenia wprzód i przeszukiwania przyrostowego z powracaniem dla obu problemów.

Algorytmy, które zaimplementowałem są generyczne. Jako parametr przyjmują problem CSP, który musi posiadać potrzebne funkcje. W ten sposób można łatwo dodać kolejne problemy CSP do programu.

1. Przeprowadzenie badania czasów przetwarzania zaimplementowanych metod oraz nakładu obliczeniowego w zależności od parametru N\* dla obu problemów.

W obu algorytmach używałem z następujących heurystyk wyboru kolejnej zmiennej/wartości:

* NORMAL – jako następna zmienna/wartość brana jest kolejna z listy dostępnych
* MEDIUM – jako następna zmienna/wartość brana jest środkowa z listy dostępnych
* MOST CONSTRAINED – jako następna zmienna/wartość brana jest najbardziej ograniczona z listy dostępnych
* LESS CONSTRAINED - jako następna zmienna/wartość brana jest najmniej ograniczona z listy dostępnych

Badania zostały przeprowadzone dla problemu n-hetmanów o n = {4, …, 20}

Porównanie Backtracking do Forward Checking z heurystyką wyboru zmiennej I wartości NORMAL:

Pomarańczowy – Forward Checking

Niebieski - Backtracking

Jak widać na wykresie, najszybszym algorytmem ze wszystkich przeze mnie przetestowanych jest Forward Checking, oraz heurystyką wyboru kolejnej zmiennej MOST CONSTRAINED.

Biorąc pod uwagę już sam algorytm Backtracking z heurystyką wyboru zmiennej MOST CONSTRAINED, najlepszą heurystyką dla wyboru kolejnej wartości również okazuje się MOST CONSTRAINED (1108 podstawień).

Czasowo wygląda to podobnie:

Porównanie Backtracking do Forward Checking z heurystyką wyboru zmiennej I wartości MOST\_CONSTRAINED:

Pomarańczowy – Forward Checking

Niebieski Backtracking

1. Wnioski.

Odpowiednie dobranie heurystyk może znacznie przyspieszyć czas działania algorytmu Forward Checking. Dla N=20 udało się zejść z 199635 podstawień w Backtracking, przez 145151 w Forward Checking bez heurystyki, aż do 144 podstawień dla heurystyki wyboru kolejnej zmiennej MOST CONSTRAINED oraz heurystyki wyboru kolejnej wartości LESS CONSTRAINED. Czasowo wygląda to analogicznie: 14 s, 81 s, 0,24 s.