Algorytmy przeszukiwania Best First Search do rozwiązania problemu n-hetmanów

Joanna Kołodziejczyk

1 Problem do rozwiązania

Celem laboratorium jest implementacja algorytmu Best First Search w celu znalezienia rozwiązania dla układanki n-hetmanów. Algorytm wykorzystuje heurystykę, tj. funkcję do szacowania jaki węzeł powinien zostać sprawdzony i rozwinięty w pierwszej kolejności.

Warunki reprezentacji problemu pozostają takie same jak w implementacji BFS i DFS celem zapewnienia spójności z poprzednim rozwiązaniem i umożliwienia porównania.

2 Algorytm Best First Search

Algorytm wykorzystuje ideę, że możliwe jest podążanie nie tylko jedną ścieżką (jak w głąb), ale przełączanie się na inną, jeżeli tylko okaże się ona lepsza od bieżącej. Ponieważ wystąpiło pojęcie „lepsza" zatem oznacza to konieczność wprowadzania funkcji oceniającej — heurystycznej.

Algorytm przeszukiwania Best First Search przeszukuje po kolei wszystkie węzły w drzewie(grafie) wybierając za każdym razem węzeł o najmniejszej wartości funkcji heurystycznej. Innymi słowy można go zaimplementować jako kolejkę priorytetową.

Funkcja heurystyczna to funkcja, której celem jest szacowanie jak daleko stan bieżący jest od dobrego rozwiązania. Funkcja nie oblicza odległości dokładnie, tylko szacuje.

Procedurę rozwiązywania n hetmanów z pomocą Best First Search można opisać następującymi krokami:

1. Utwórz pustą kolejkę tablic/wektorów na listę stanów do przeszukania
2. Dodaj do listy stan początkowy
3. W pętli wykonuj dopóki lista stanów do przeszukania nie jest pusta
   (a) Pobierz z **początku** listy stan (stan o najmniejszej wartości f. heurystycznej)
   (b) Sprawdź warunek osiągnięcia celu. Jeżeli został osiągnięty możesz przerwać procedurę i wyświetlić pierwsze znalezione rozwiązanie (opcjonalnie można kontynuować do znalezienia wszystkich rozwiązań). W przeciwnym wypadku kontynuuj.
   (c) Wygeneruj wszystkich potomków aktualnego stanu, oszacuj dla nich wartość funkcji heurystycznej i dodaj ich do listy stanów do przeszukiwania w kolejności rosnącej wartości funkcji heurystycznej.

3 Heurystyki

Funkcja heurystyczna powinna być dodatnią funkcją malejącą wraz z rosnącą szansą na rozwiązanie. Przyjmuje się, że rozwiązanie powinno mieć wartość 0. W przypadku rozważanych poniżej heurystyk nie zawsze ten warunek jest spełniony.

3.1 Heurystyka H1

Motywacja: Wybór jako pierwszych hetmanów ze środkowych wierszy. Bardzo często na brzegach pojawiają się niewłaściwe rozwiązania.

Załóżmy, że n - liczba hetmanów, i - liczba hetmanów już stojących na szachownicy, w_{row} - waga numeru wiersza, do którego jest wstawiany hetman row_i obliczana jako:

$$w_{row} = \begin{cases} n - row_i + 1 & \text{if } row_i < n/2 \\ row_i & \text{otherwise} \end{cases}$$

Wartość funkcji heurystycznej może mieć postać:

$$h_1 = (n - i) \cdot \sum_{j=1}^{i} w_{row_j}$$

,

Interpretacja: Każdy węzeł jest oceniany liczbą wstawionych już hetmanów $(n - i)$, oraz sumą wag wierszy, w których stoją już wstawieni hetmani.

3.2 Heurystyka H2

Motywacja: Wybierać takie węzły, gdzie liczba dopuszczalnych pozycji (pól, które nie są jeszcze atakowane przez wstawionych już hetmanów) jest największa.

Należy zatem brać pod uwagę, jakie pola są już na szachownicy atakowane. Innymi słowy chcemy wybierać takie węzły, w których liczba pól zagrożonych biciem jest jak najmniejsza, ich liczbę oznaczamy jako $N_{infeasible}$.

Wartość funkcji heurystycznej może mieć postać:

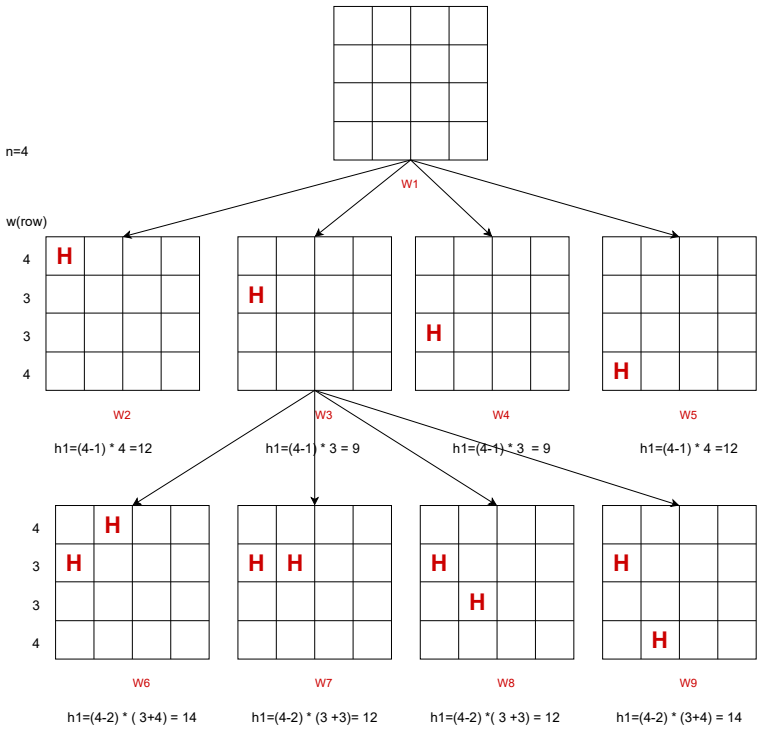
$$h_2 = N_{infeasible}$$

Interpretacja: Każdy węzeł jest oceniany liczbą pól atakowanych $N_{infeasible}$.

3.3 Heurystyka H3

Motywacja: Skumulowana odległość Hamminga pomiędzy wszystkimi hetmanami na szachownicy ma pozwalać na dobrą separację. Chcemy, by odległość była maksymalna.

Odległość Hamminga d_H dla pewnego stanu jest obliczana w następujący sposób: Dla hetmana $Q1$ sprawdzamy, czy wartość jest różna od hetmana $Q2$. Jeżeli tak, to $d_H(Q1 : Q2) = 1$, w przeciwnym przypadku jest równa 0. Następnie jeżeli jest wstawiony hetman $Q3$ to badamy, czy indeksy są różne pomiędzy $Q1$ i $Q3$. Potem zliczam czy $Q2$ ma różne wartości indeksów od $Q3$ i kolejnych. itd.



Rysunek 1: Ilustracja heurystyki H1

Jeżeli maksymalna liczba różnic w pozycjach każdego hetmana z każdym hetmanem to

$$S = \frac{n}{2} \cdot (n-1)$$

Wartość funkcji heurystycznej może mieć postać:

$$h_3 = S - d_H$$

Interpretacja: Obliczmy d_H zgodnie z algorytmem podanym wyżej i odejmujemy od maksymalnej liczby d_H dla problemu n-hetmanów, czyli S. Uzyskanie wartości $h_3 = 0$ nie gwarantuje rozwiązania.

3.4 Heurystyka H4

Motywacja: Odległość Manhattan pomiędzy wszystkimi hetmanami powinna być jak najbliższa wielkości 3 (separacja ruchem skoczka szachowego). Chcemy zatem preferować takie układy, które taką separację mają.

Zapis funkcji pozostawiam do samodzielnego opracowania.

4 Wymagania dla implementacji

Język implementacji taki sam jak w przednich zadaniach. Wszelkie założenia dotyczące reprezentacji problemu pozostają takie same.

Celem zadania programistycznego jest porównanie skuteczności algorytmu Best First Search z różnymi heurystykami z DFS i BSF.

Zadania:

1. Zaprogramować algorytm Best First Search dla dwóch wybranych heurystyk (można zaproponować własne).
2. Program po wykonaniu powinien wyświetlać:
   - stan, który został wskazany przez program jako rozwiązanie, czyli współrzędne hetmanów na planszy (możliwe jest użycie wizualizacji),
   - statystyki:
     - liczbę stanów wygenerowanych,
     - liczbę stanów sprawdzonych (dla których wykonany został test osiągnięcia celu),
     - względny czas wykonania.
3. Wykonać eksperymenty obliczeniowe dla n zmieniającego się od 4 do np. 20.
4. Zademonstrować wyniki eksperymentu w postaci wykresu zmienności statystyk w zależności od liczby hetmanów. Na osi odciętych powinny być przyrastające wartości n a na osi rzędnych wartości statystyk.
5. Połączyć wyniki ze statystykami z poprzednich zadań.

4.1 Przekazanie programu

- Kod z rozwiązaniem proszę podpiąć w Teams. Bardzo proszę nazwać plik źródłowy nazwisko.imię.???.
- Wykresy ze statystykami proszę załączyć też w Teams.