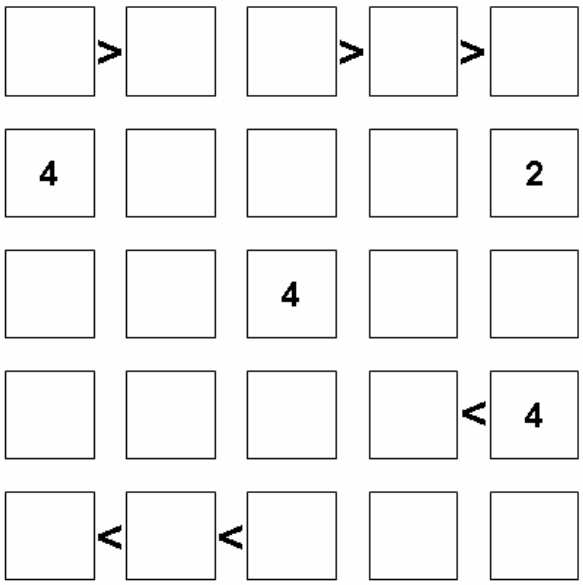
Rozwiązywanie planszy Futoshiki przy użyciu IBM ILOG CPLEX

Aleksandra Walczybok 272454
Hanna Zwolińska 272440

8 czerwca 2024

1 Wstęp

Projekt dotyczy rozwiązania problemu logicznego Futoshiki, który jest grą podobną do Sudoku, ale z dodatkowymi ograniczeniami relacyjnymi. Gra rozgrywa się na planszy $n \times n$, gdzie celem jest umieszczenie liczb od 1 do n w taki sposób, aby każda liczba pojawiła się dokładnie raz w każdym wierszu i kolumnie. Dodatkowo, pewne komórki planszy są powiązane relacjami "większe niż" lub "mniejsze niż", które muszą być spełnione.



Rysunek 1: Przykładowa plansza do gry.

2 Wczytywanie danych

Plik danych `futoshiki.dat` zawiera definicje zmiennych n, *initialValues* oraz *conditions*. Każda z tych zmiennych jest wypełniona odpowiednimi wartościami zgodnie z wymaganym formatem.

- n (`int`) - wymiar planszy
- *initialValues* (`set`) - zadane wartości początkowe podawane w `tupli` o postaci: <współrzędna x, współrzędna y, wartość>
- *conditions* (`set`) -zadane warunki pomiędzy komórkami podawane w `tupli` o postaci: <współrzędna x1, współrzędna y1, współrzędna x2, współrzędna y2, warunek ("<" lub ">")>

Dane zostają wczytane i zapisane przy użyciu `tupli` o nazwach i strukturach:

- *Constraint* (`tuple`) - <`int` x1, `int` y1, `int` x2, `int` y2, `string` relation>
- *Cell* (`tuple`) - <`int` row, `int` col, `int` value>

Zapisanie danych w ten sposób ułatwia zapis ograniczeń.

3 Ograniczenia użyte do rozwiązania problemu

Inicjalizacja wartości w komórkach (*initialValues*)

$$\forall\, \text{cell} \in \text{initialValues},\ \text{jeśli}\ \text{cell.value} \neq 0,\ \text{to}\ x[\text{cell.row}][\text{cell.col}] = \text{cell.value} \tag{1}$$

Unikalność w wierszach

$$\forall\, i \in \{1,\ldots,n\},\ \forall\, k1 \in \{1,\ldots,n\},\ \forall\, k2 \in \{k1+1,\ldots,n\}, \quad x[i][k1] \neq x[i][k2] \tag{2}$$

Unikalność w kolumnach

$$\forall\, j \in \{1,\ldots,n\},\ \forall\, k1 \in \{1,\ldots,n\},\ \forall\, k2 \in \{k1+1,\ldots,n\}, \quad x[k1][j] \neq x[k2][j] \tag{3}$$

Ograniczenia relacyjne

$$\forall\, c \in \text{conditions} \begin{cases} x[c.x1][c.y1] > x[c.x2][c.y2] & \text{jeśli}\ c.\text{relation} = \text{" > "} \\ x[c.x1][c.y1] < x[c.x2][c.y2] & \text{jeśli}\ c.\text{relation} = \text{" < "} \end{cases} \tag{4}$$

4 Algorytm heurystyczny

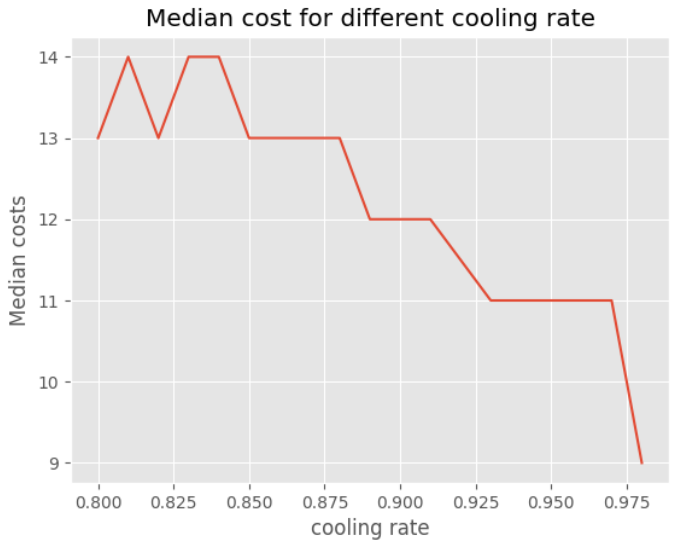
Wybranym algorytmem heurystycznym jest symulowane wyżarzanie. Został on zaimplementowany w Pythonie. Celem algorytmu jest wybranie takiej konfiguracji cyfr na planszy, aby jak najlepiej spełnione zostały zadane ograniczenia. Poprawność jest sprawdzana poprzez wartość funkcji kosztu (chcemy aby była ona jak najmniejsza). Funkcja ta jest obliczana na następującej podstawie: za każde powtórzenie cyfry w wierszu oraz kolumnie dodawane jest +1 pkt. Dodatkowo za każde niespełnione ograniczenie relacyjne dodawany jest kolejny punkt. Funkcje kosztu w zależność od parametrów symulowanego wyważania (temperatura oraz współczynnik schładzania) pokaze są na poniższych wykresach.

4.1 Przykładowy wynik działania funkcji symulowanego wyżarzania.

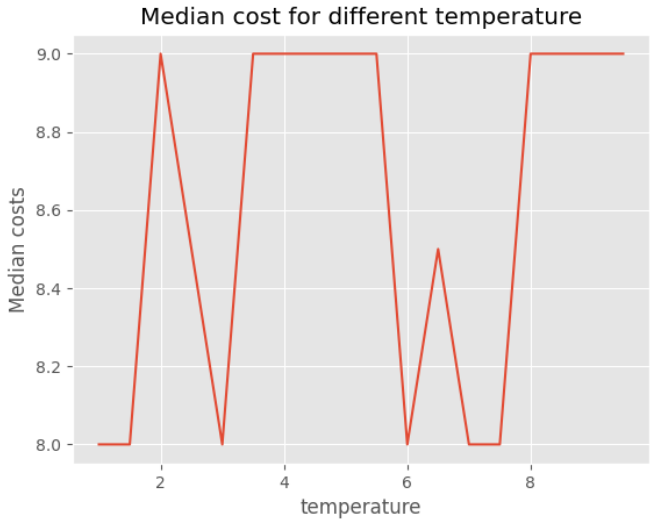
```
constraint1 = [[(0, 0), (0, 1), ">"],
               [(1, 3), (1, 4), "<"],
               [(3, 1), (3, 2), "<"]]

Best matrix:
[[5 2 4 1 2]
 [1 3 2 2 5]
 [5 2 1 4 3]
 [2 1 5 5 2]
 [5 5 2 3 1]]
Best cost: 10
Iterations: 228
```

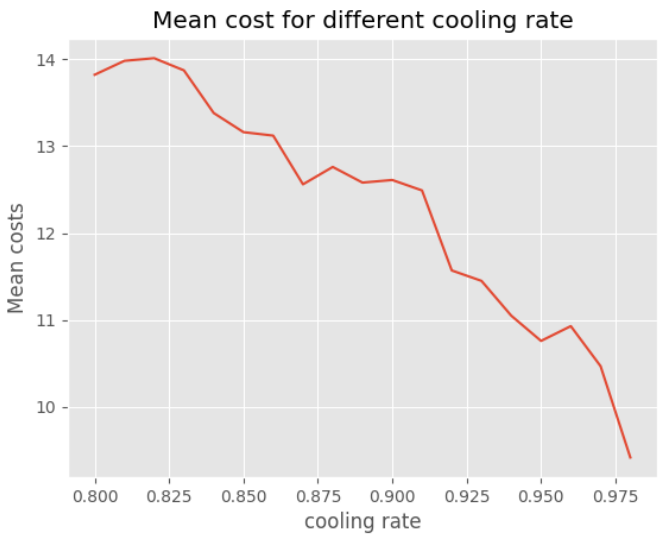
4.2 Wykresy jako wynik działania symulowanego wyżarzania.



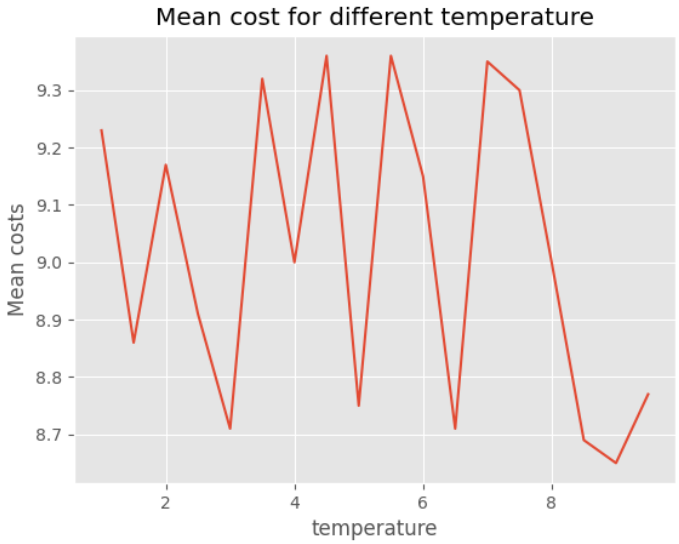
Rysunek 2: Wykres mediany kosztu od współczynnika schładzania.



Rysunek 3: Wykres mediany kosztu od temperatury.



Rysunek 4: Wykres średniej wartości kosztu od współczynnika schładzania.



Rysunek 5: Wykres średniej wartości kosztu od temperatury.

5 Wnioski

Jak można zauważyć na powyższych przykładach, symulowane wyżarzanie nie radzi sobie za dobrze z tym problemem optymalizacji. Mimo wielu iteracji oraz prób zmieniania parametrów koszt nie schodził poniżej wartości 6. Świadczy to więc o tym, iż wybrany program, w którym pierwotnie implementowałyśmy rozwiązanie jest lepszym wyborem w celu rozwiązania tego problemu.

6 Załączniki

Link do kodu źródłowego: GitHub.