Sprawozdanie z implementacji metaheurystki "Ant Colony Optimization"

Miłosz Chodkowski 141144
milosz.chodkowski@student.put.poznan.pl

18 stycznia 2020

Spis treści

1 Treść problemu

Dany jest graf spójny ze zbiorem wierzchołków V i zbiorem krawędzi E oraz wagami w_i przypisanymi do każdej krawędzi. Należy znaleźć ścieżkę łączącą wszystkie wierzchołki (każdy odwiedzony min. raz) taką, aby minimalizować jej koszt S. S obliczany jest w taki sposób, że stanowi ona sumę wszystkich wag krawędzi licząc od tej wychodzącej z pierwszego wierzchołka ścieżki do wagi krawędzi wchodzącej do ostatniego wierzchołka ścieżki, w taki jednak sposób, że jeśli właśnie dodawana do S waga wi jest większa niż następna waga krawędzi na tej ścieżce: (tj. niż waga w_{i+1}), wtedy do sumy dodajemy nie samo w_i, ale jej 10-krotność (czyli 10 * w_i).

2 Opis metaheurystyki

Wykorzystywane biblioteki zewnętrzne:

- NumPy (zwiększa szybkość programu, poprawia czytelność kodu)

Algorytm jest oparty na na naturalnych zachowaniach kolonii mrówek. Działa na 3 głównych obiektach, Graph, Ant i obiekcie ACO, który reprezentuje całą metaheurystykę. Ma on szereg parametrów sterujących:

- Liczba iteracji I.
- Rozmiar kolonii mrówek σ.
- Wpływ feromonu i dystansu na decyzję mrówki α oraz β.
- Intensywność feromonu (ilość pozostawianego feromonu) ψ.
- Procent odparowywanego feromonu ρ.

Metaheurystyka ta jest algorytmem iteracyjnym, z każdą iteracją stara się on poprawić poprzednie rozwiązanie. Każda iteracja rozpoczyna się stworzeniem kolonii mrówek (obiekty Ant), które wykonują operacje na wcześniej stworzonym lub wczytanym z pliku Grafie (obiekt Graph). Każda komórka w macierzy tego grafu zawiera dystans od wierzchołka i do wierzchołka j. Wartości inf oznaczają brak krawędzi między tymi wierzchołkami. Program sprawdza czy graf jest grafem spójnym przez wykonanie na nim algorytmu DFS. Jeśli algorytm przeszedł przez $|V|$ wierzchołków, to znaczy, że graf jest spójny. Dodatkowo zostaje stworzona macierz feromonowa dla tego grafu. Każda komórka jest wypełniona wartościami $(\frac{1}{|V|/2})^2$

W trakcie iteracji każda z mrówek dostaje losowy wierzchołek startowy. Następnie dla każdego wierzchołka z nim połączonego jest generowane prawdopodobieństwo na podstawie macierzy feromonowej oraz wybór następnego wierzchołka na podstawie tych prawdopodobieństw (jeśli jest to pierwsza iteracja, to wybór jest losowy; w pierwszej iteracji każdy wybór ma takie samo prawdopodobieństwo).

Każdy wybór (każde przejście do innego wierzchołka) jest zapisywane do 2 list. Jedną jest listą odwiedzonych wierzchołków (zapisywana jest w niej ścieżka jaką pokonała mrówka). Wg problemu, jeśli następna krawędź na trasie jest dłuższa od poprzedniej, to z całkowitego kosztu ścieżki odejmowana jest waga poprzedniej krawędzi, a następnie dodawana ponownie, lecz powiększona 10 razy. Drugą jest lista ruchów tabu (lista ruchów, których nie można wykonać). Do listy tabu dodawany jest każdy odwiedzony wierzchołek. W trakcie wybierania nowego wierzchołka każda mrówka zagląda do listy tabu i szuka w niej kandydata na następny ruch. Jeśli kandydat jest obecny w liście, to nie może zostać wybrany jako następny wierzchołek i wybierany jest kolejny kandydat. Jeśli dojdzie do sytuacji w której brak jest kandydata na następny wierzchołek, to zostaje uwolnione 25% ruchów w liście tabu i generowanie następnego ruchu rozpoczyna się od nowa do czasu aż znajdziemy potencjalnych kandydatów na następny wierzchołek (list tabu jest to zabezpieczenie przed niepotrzebnymi nawrotami mrówek do poprzednich wierzchołków).

Sam wybór odpowiedniego wierzchołka jest dokonywany przez odpowiednią funkcję numpy'a. Dla do listy z kandydatami na wierzchołek mapowana jest funkcja generująca dla każdego z nich prawdopodobieństwo. Nastepnie do funkcji numpy.choice zostaje podana lista z kandydatami oraz lista z odpowiednimi prawdopodobieństwami wyboru tego kandydata. Funckja z odpowiednim prawdopodobieństwem wybiera następny wierzchołek. Problemem tutaj jest to, że w pewnym momencie wartości prawdopodobieństwa stają się ekstremalnie niskie. Nawet język Python nie jest w stanie ich obsłużyć i są one zapisywane jako NaN (Not a Number). Aby temu zapobiec, po każdej generacji prawdopodobieństwa zostaje wywołana metoda sprawdzająca poprawność listy prawdopodobieństw dla aktualnego wierzchołka. Jeśli wartości stają się ekstremalnie małe (NaN), to zostają one najpierw zmienione na najniższe float'y które może obsłużyć system, a następnie cała lista zostaje pomnożona $*= 1.1^{|V|}$. W kolejnym kroku wszystkie liczby zostają podzielone przez sumę całej listy, co sprawia że prawdopodobieństwa sumują się do 1.0.

Każda z mrówek powtarza tę sekwencję czynności do czasu aż liczba unikalnych odwiedzonych wierzchołków będzie równa $|V|$. Jeśli któraś z mrówek będzie „podróżować" przez graf więcej niż 5 sekund, wtedy program używa break'a do wyjścia z pętli i rozpoczyna obliczenia dla kolejnych mrówek (ta

opcja nie jest zawsze ustawiona na 5 sekund. Jest to zależne od czynników takich jak rozmiar instancji czy ilość mrówek. W testach ten czas był zmieniany w taki sposób, aby średnio 90% mrówek mogło znaleźć rozwiązanie). Następnie rozwiązanie znalezione przez mrówkę jest porównywane z najlepszym rozwiązaniem. Jeśli jest ono lepsze, to mrówka zostaje dodana do listy najlepszych mrówek (te, które polepszyły rozwiązanie) oraz jej rozwiązanie zostaje przypisane jako nowe najlepsze rozwiązanie. Po zakończeniu czynności wszystkich mrówek, na każdą odwiedzoną przez najlepsze mrówki krawędź zostaję nałożony feromon w ilości:

$$(\frac{\psi}{S})^2 \ : \ S = \text{koszt ścieżki}$$

W ostatnim kroku zostaje odparowana część feromonu na wszystkich krawędziach i rozpoczyna się kolejna iteracja.

3 Istotne wyjątki

W trakcie testów działania algorytmu napotkano wiele problemów. Jednym z głównych było wpadanie mrówek w lokalne optimum (np. cykl lub naprzemienne przechodzenie przez wierzchołki ***i, j, i, j, i ...*** Prostym zabezpieczeniem przed taką sytuacją jest lista tabu opisana w akapicie powyżej.

Kolejnym problemem była zbyt duża dominacja jednego rozwiązania nad innymi. Mrówki po znalezieniu dobrego rozwiązania bardzo rzadko wykorzystywały ścieżki niepołączone w żaden sposób z najlepszym dotychczasowym rozwiązaniem. Sposobem na przynajmniej częściowe uniknięcie tego problemu było wygładzenie macierzy feromonowej w konkretnych sytuacjach (testy na końcu sprawozdania).

Jednym z ostatnich problemów był problem zbyt długiego przechodzenia grafu przez mrówki. W niektórych przypadkach (najczęściej przy małych grafach) przechodziły one bardzo długo przez graf. Taka sytuacja zabierała cenny czas, a nie dawała żadnego rozwiązania. W tym celu każda z mrówek posiadała licznik czasu. Jeśli czas przekroczył podany limit (dobierany w zależności od rozmiaru grafu), to „porzucano" obecną mrówkę oraz jej rozwiązanie, a następna mrówka rozpoczynała przechodzenie grafu. Algorytm posiadał również parametr który określał jak wiele mrówek może zostać porzuconych bez określonej reakcji. Parametr ten (nazwany ***break_count***) umożliwia wygładzenie macierzy feromonowej, jeśli zbyt wiele mrówek zostaje porzuconych (jeśli licznik porzuconych mrówek > ***change_count***). W wielu przypadkach taki zabieg pozwolił na zaoszczędzenie kilku sekund cennego czasu (przy dużych grafach z małą ilością możliwych rozwiązań były

to już minuty). Niestety wpłynęło to negatywnie na wartość funkcji celu. Przy grafach z małą ilością możliwych rozwiązań zbyt wiele mrówek było „porzucanych". Skutkiem tego była mała ilość polepszeń.

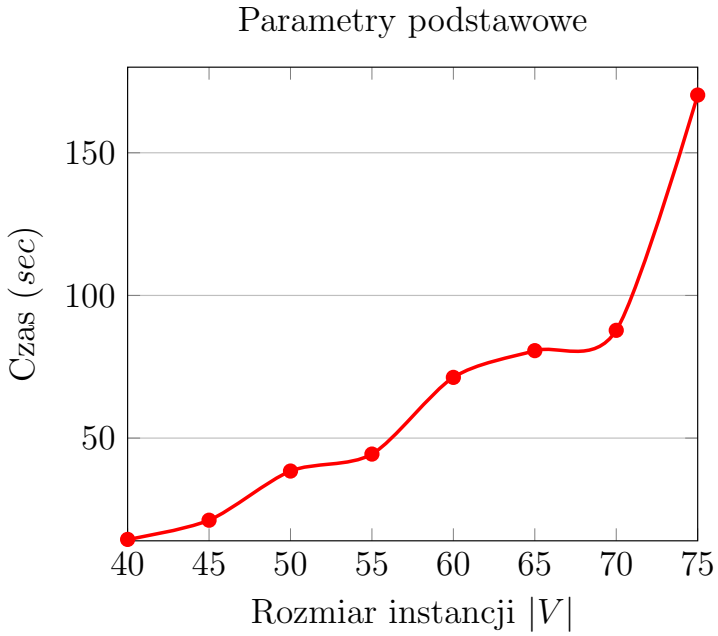
4 Testy parametrów

Testy algorytmu zostały przeprowadzone na 5 instancjach (po 5 grafów na rozmiar). Grafy spójne o rozmiarach $|V|$ odpowiednio: 40, 45, ..., 70. Każdy test został przeprowadzony 5 razy, wyniki są średnią obliczoną z 5 prób lub najlepszym wynikiem uzyskanym w ciągu testów.

Podstawowe parametry algorytmu:

- Liczba mrówek = 10
- Liczba iteracji = 50
- Wpływ feromonu = 0.45
- Wpływ dystansu = 0.55
- Ilość odparowanego feromonu = 0.5
- Intensywność feromonu (ilość zostawianego feromonu) = 1.0

4.1 Wpływ rozmiaru instancji na czas obliczeń

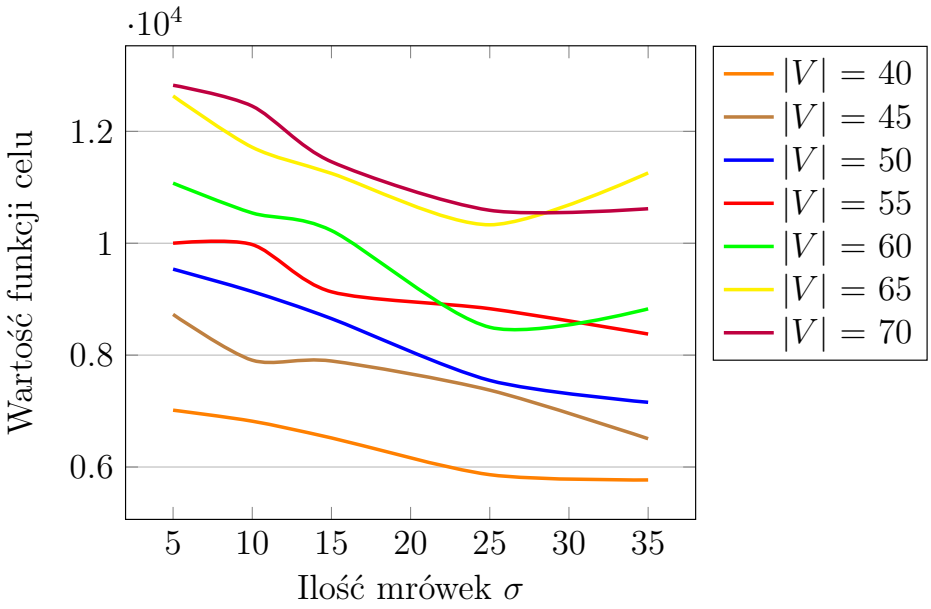


Wnioski:

Z powyższego wykresu wynika, że wraz z rozmiarem instancji rośnie czas optymalizacji. Jest to związane ze zwiększoną ilością obliczeń. Każda mrówka musi obliczyć prawdopodobieństwo dla swojego sąsiedztwa, następnie przejść do kolejnego wierzchołka, ponowić obliczenie prawdopodobieństwa itd. Jest to proces bardzo czasochłonny, ponieważ użyte w programie struktury mają dość wysoki czas dostępu, średnio $O(n^2)$. Powyżej rozmiaru 70 czas obliczeń drastycznie wzrasta, dlatego do testów użyto grafów o rozmiarach $<= 70$
.

4.2 Wpływ ilości mrówek na najniższą otrzymaną wartość funkcji celu

$I = 50$, $\alpha = 0.45$, $\beta = 0.55$, $\rho = 0.5$, $\phi = 1.0$

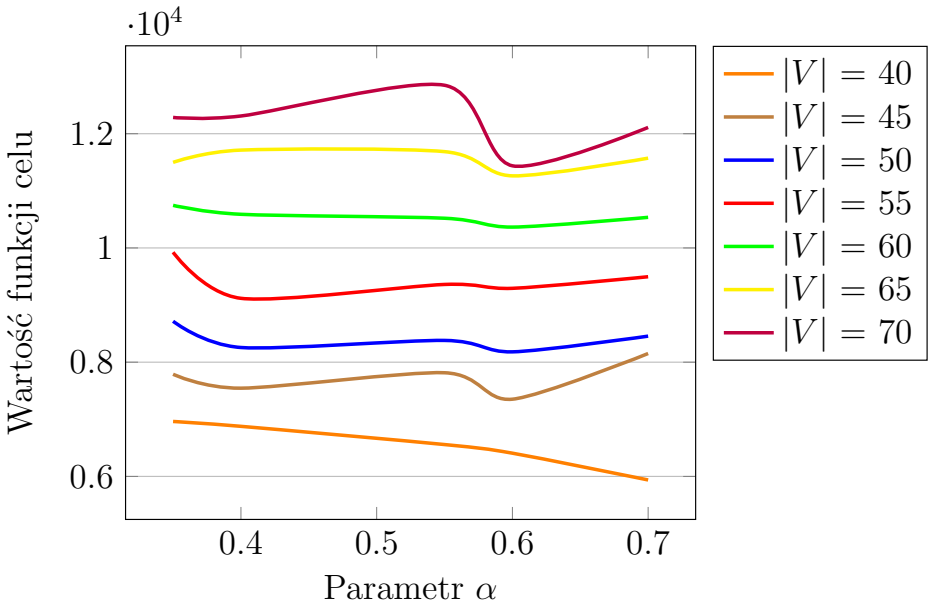


Wnioski:

Jak wynika z powyższego wykresu – w każdej instancji średnio najniższa otrzymana wartość funkcji celu maleje wraz ze wzrostem ilości mrówek poszukujących rozwiązanie. Jest to skutek tego, że większa liczba mrówek pozwala na otrzymanie większej liczby rozwiązań, dzięki czemu szansa na to, że otrzymane rozwiązanie będzie lepsze od swojego poprzednika jest większe.

4.3 Wpływ parametru α na najniższą otrzymaną wartość funkcji celu

$I = 50$, $\sigma = 10$, $\beta = 0.55$, $\rho = 0.5$, $\phi = 1.0$

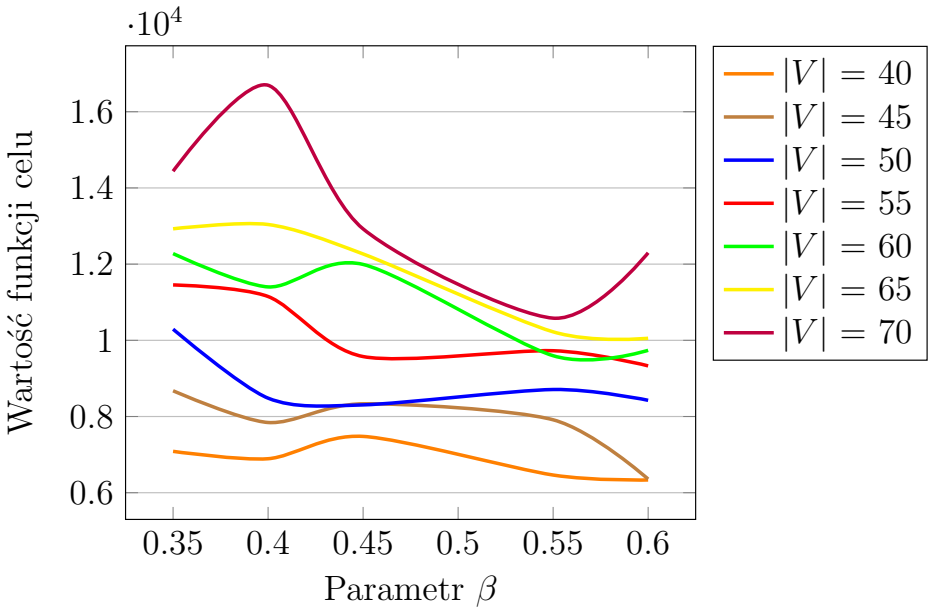


Wnioski:

Wpływ feromonu jest parametrem który działa najlepiej, jeśli przyjmuje wartości tylko małe lub tylko duże. Optimum zależy głównie od rodzaju testowanej instancji. Jego dwa lokalne optima znajdują się w okolicach wartości 0.4 oraz 0.6.

4.4 Wpływ parametru β na najniższą otrzymaną wartość funkcji celu

$I = 50$, $\sigma = 10$, $\alpha = 0.45$, $\rho = 0.5$, $\phi = 1.0$

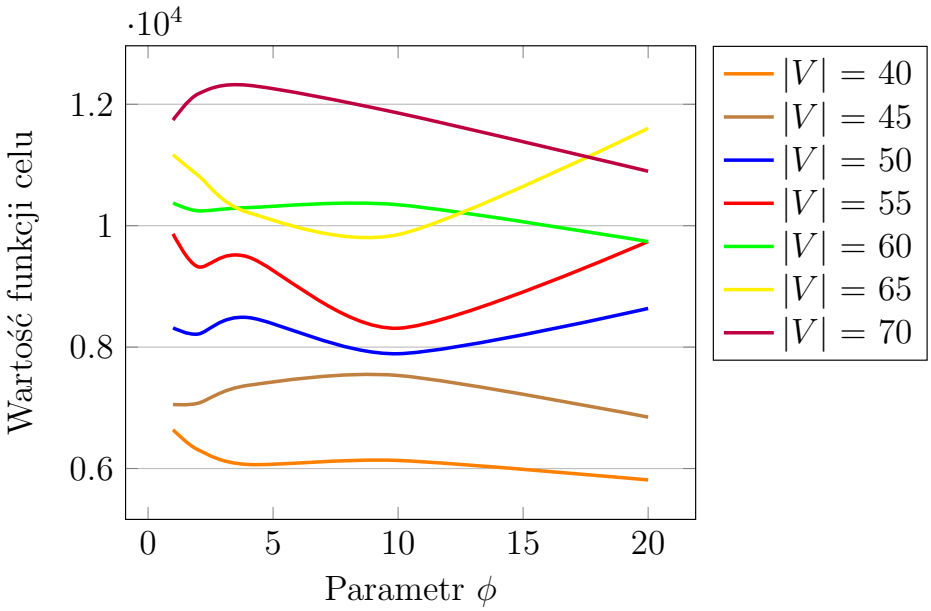


Wnioski:

Wpływ dystansu jest parametrem w którym bardzo dobrze widoczne jest lokalne optimum. Algorytm działa najlepiej jeśli $\beta \in [0.55, 0.6]$. Wynika to z tego, że duży dystans między wierzchołkami i, j drastycznie zmniejsza prawdopodobieństwo wyboru długiej krawędzi przez mrówkę, dzięki czemu mrówki częściej wybierają krawędzie w sposób podobny do losowego.

4.5 Wpływ parametru ϕ na najniższą otrzymaną wartość funkcji celu

$I = 50$, $\sigma = 10$, $\alpha = 0.45$, $\beta = 0.55$, $\rho = 0.5$



Wnioski:

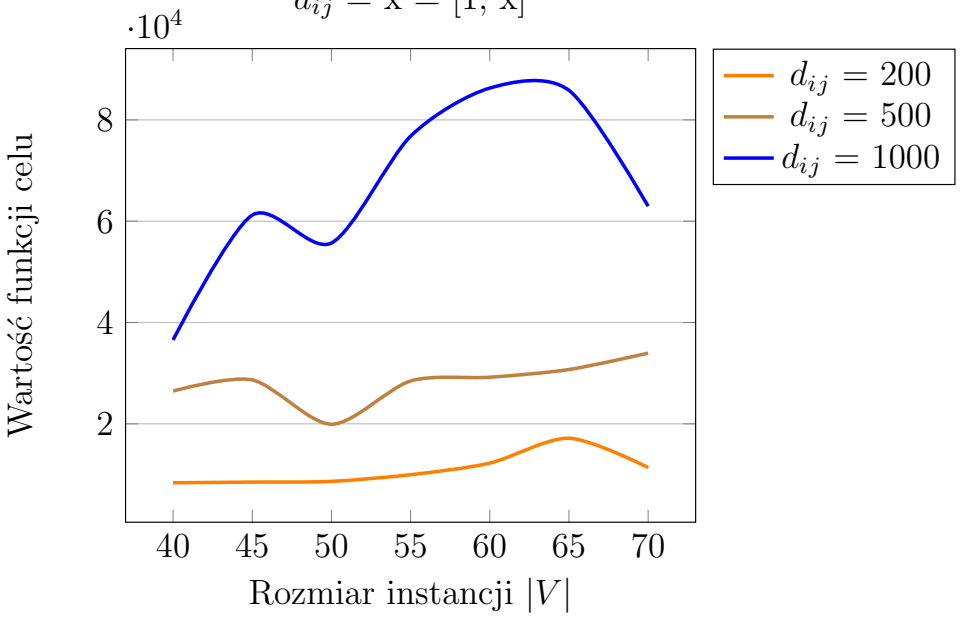
Intensywność feromonu jest jednym z ważniejszych parametrów. Dzięki odpowiedniemu ustawieniu tego parametru możemy bezpośrednio wpłynąć na wartość funkcji celu. Na powyższym wykresie widzimy, że lokalne optimum znajduje się w przedziale $\phi \in [8, 14]$. Taka wartość mogła wpłynąć pozytywnie na wartość funkcji celu, ponieważ np. zwiększa ona szansę na ponowne wybranie krawędzi 'dobrej', tzn. takiej, która należy do dobrego rozwiązania. Z takich krawędzi mrówki mogą się 'rozpraszać' do innych. Ponowne wybieranie krawędzi należących do jakiegoś dobrego rozwiązania zmniejsza nam szanse na to, że mrówka znajdzie rozwiązanie gorsze od obecnego.

5 Testy instancji

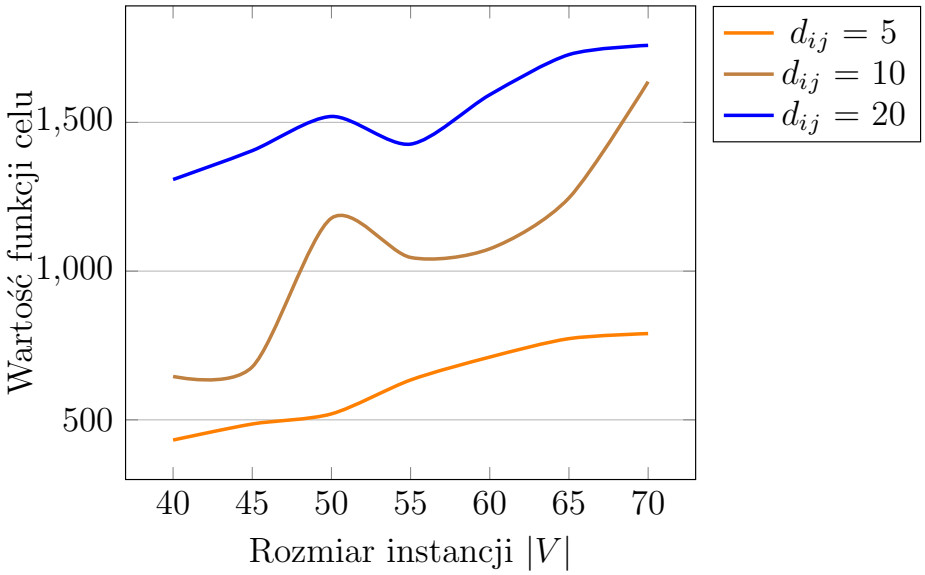
Testy zostały przeprowadzone na jednakowych instancjach ze zmodyfikowanymi długościami krawędzi (układ krawędzi pozostał taki sam). Tak jak w testach parametrów każda instancja została poddana testowi 5 razy. Wyniki są średnią z najniższych uzyskanych wyników w ciągu tych 5 testów.

5.1 Wpływ długości krawędzi d_{ij} na najniższą otrzymaną wartość funkcji celu

$I = 50$, $\sigma = 10$, $\alpha = 0.45$, $\beta = 0.55$, $\rho = 0.5$, $\phi = 1.0$
$d_{ij} = \mathrm{x} = [1, \mathrm{x}]$



$I = 50$, $\sigma = 10$, $\alpha = 0.45$, $\beta = 0.55$, $\rho = 0.5$, $\phi = 1.0$
$d_{ij} = \mathrm{x} = [1, \mathrm{x}]$



Wnioski:

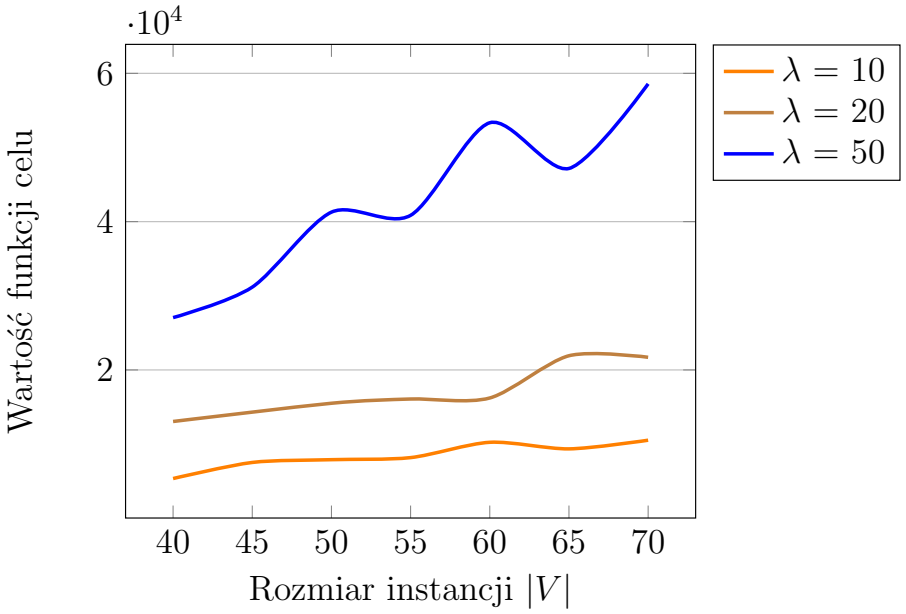
Rozmiar instancji ma realny wpływ na uzyskaną wartość funkcji celu. Odmienne od podstawowych koszty krawędzi w grafie powodują że mrówki:

- Pozostawiają mniej lub więcej feromonu na krawędziach (w zależności od wagi. Dla dużych wag - mniej, dla małych - więcej).
- Swoją decyzję prawie całkowicie uzależniają od wagi krawędzi. Feromon na niej pozostawiony staje się jedynie "dodatkiem", który może na nie wpłynąć tylko przy dużych wartościach.

Podsumowując, im większe wagi krawędzi, tym trudniej o uzyskanie większego polepszenia wartości funkcji celu.

5.2 Wpływ kary λ na najniższą otrzymaną wartość funkcji celu

$I = 50$, $\sigma = 10$, $\alpha = 0.45$, $\beta = 0.55$, $\rho = 0.5$, $\phi = 1.0$



Wnioski:
Wpływ kary oprócz podniesienia wartości funkcji celu, spowodował również mniejszą ilośc polepszeń rozwiązania (nie umieszczone na wykresie). Oznacza to, że podniesienie kary ma bardzo negatywny wpływ na działanie algorytmu.

6 Testy dodatkowe

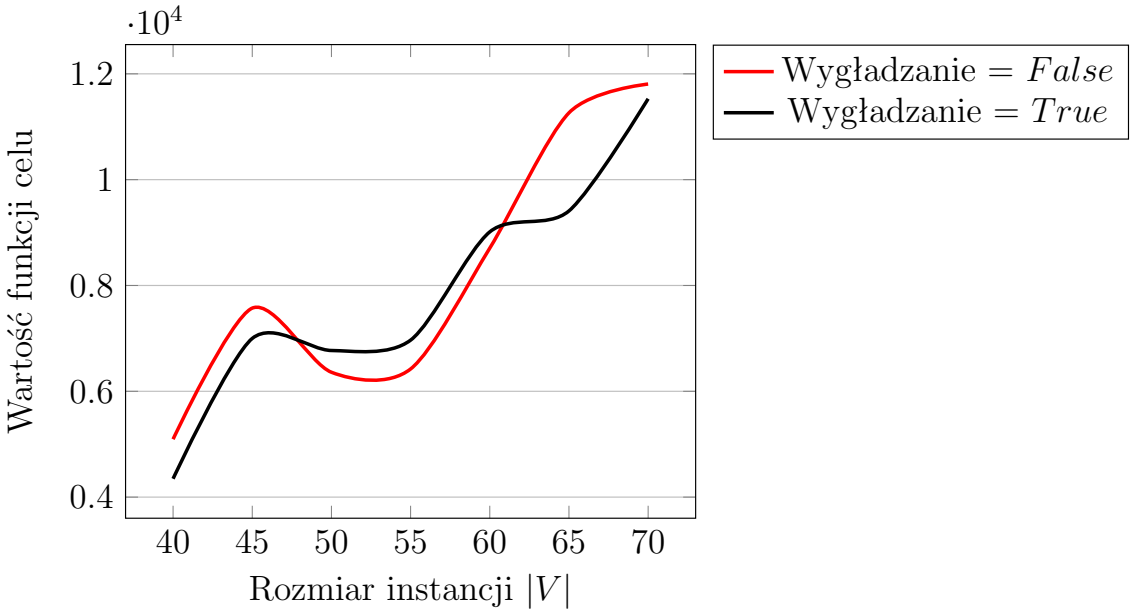
Sposób wygładzania krawędzi:

$$\tau_{ij} = \tau_{min} * \sqrt{\frac{\tau_{ij}}{\tau_{min}}}$$

- τ_{ij} = feromon na krawędzi i, j
- τ_{min} = min(macierz feromonowa)

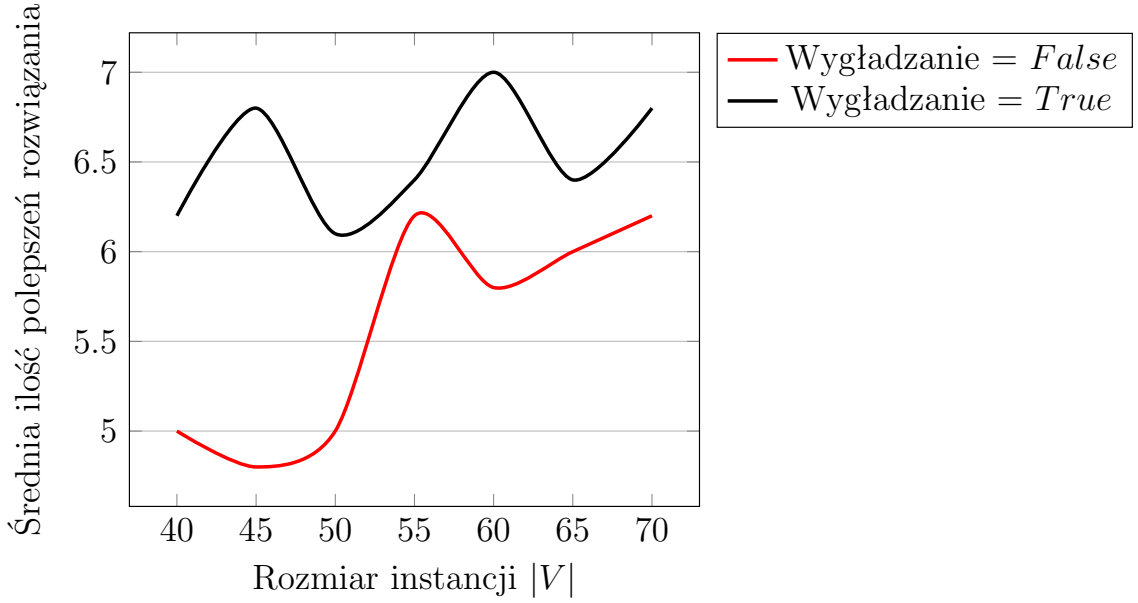
6.1 Wpływ wygładzania macierzy na najniższą uzyskaną wartość funkcji celu

$I = 50$, $\sigma = 10$, $\alpha = 0.45$, $\beta = 0.55$, $\rho = 0.5$, $\phi = 1.0$



6.2 Wpływ wygładzania macierzy na średnią ilość polepszeń rozwiązania

$I = 50$, $\sigma = 10$, $\alpha = 0.45$, $\beta = 0.55$, $\rho = 0.5$, $\phi = 1.0$



Wnioski:

Wygładzanie macierzy jest dobrym sposobem neutralizacji dominacji niektórych krawędzi nad innym. W niektórych przypadkach pozwala ono nie tylko uzyskać lepsze rozwiązanie, ale także więcej razy poprawić rozwiązanie poprzednie.

7 Podsumowanie

Realizacja tego projektu była bardzo pouczająca. Dała mi podstawowy przegląd technik optymalizacji kombinatorycznej. Dzięki zaimplementowaniu tego algorytmu doszedłem do wniosku, że tworzenie jak i implementacja metaheurystyk jest trudna i wymaga wiele wiedzy. Podsumowując, wykonanie projektu było ciekawym doświadczeniem, jednak niestety nie udało mi się go rozwinąć w idealny sposób. Uważam, że byłem w stanie zrobić więcej i myślę, że zdobytą wiedzę na pewno wykorzystam w przyszłości.