Sterowanie procesami dyskretnymi
Tabu Search

Mateusz Górka
Karolina Głuszek
Michał Kowalski

21 kwietnia 2021

1 Wstęp i wyniki pomiarów

Przygotowana przez nas implementacja metaheurystyki posiada szereg modyfikacji wpływających na działanie algorytmu. Są to kolejno:

1. Długość tablicy tabu,
2. Podalgorytm do wyznaczania rozwiązań,
3. Sposób zamiany danych (zamiana, wstawienie, odwrócenie kolejności),
4. Kryterium (lub kryteria) zakończenia działania (ilość iteracji, czas, ilość iteracji bez poprawy rozwiązania).

Wybierając 3 różne długości tablicy tabu, osiem zaimplementowanych podalgorytmów, 3 sposoby zamiany danych, trzy kryteria zakończenia dla trzech różnych wartości i na trzech zestawach danych, oraz wliczając mieszanie kryteriów zakończenia musielibyśmy przeprowadzić ilość testów równą:

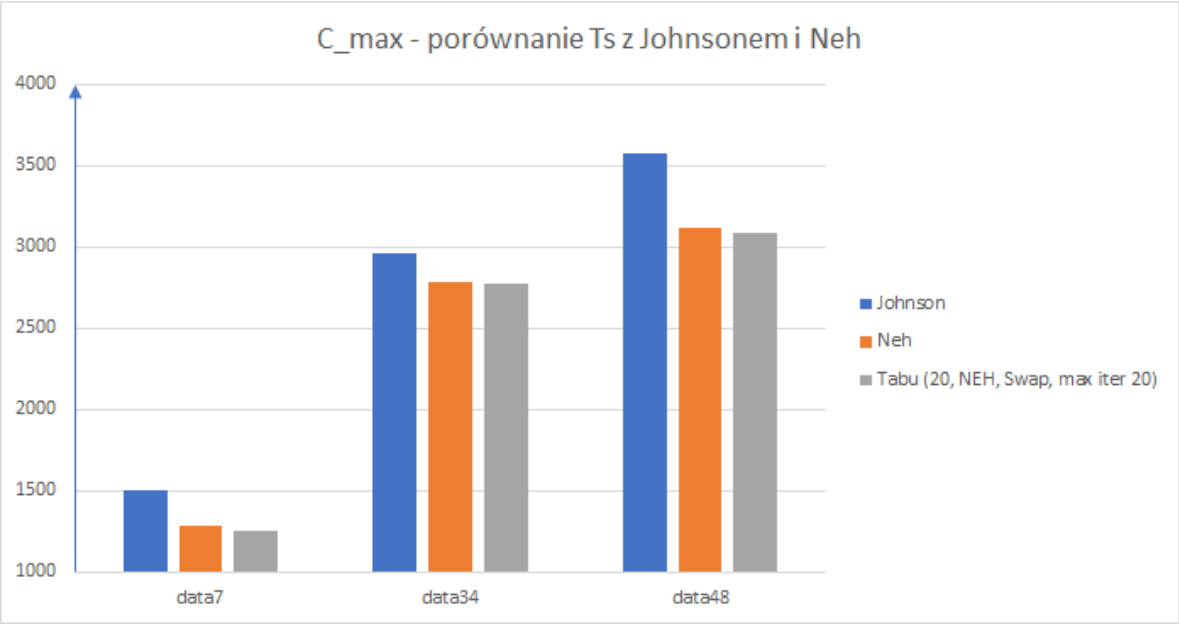
$$3^5 \cdot 8 \cdot 6 = 11664$$

Z tego względu, że nie jest to możliwe, przeprowadzone testy mogą być niedokładne, ponieważ przetestowana została stosunkowo niewielka liczba kombinacji różnych modyfikacji Tabu searcha.

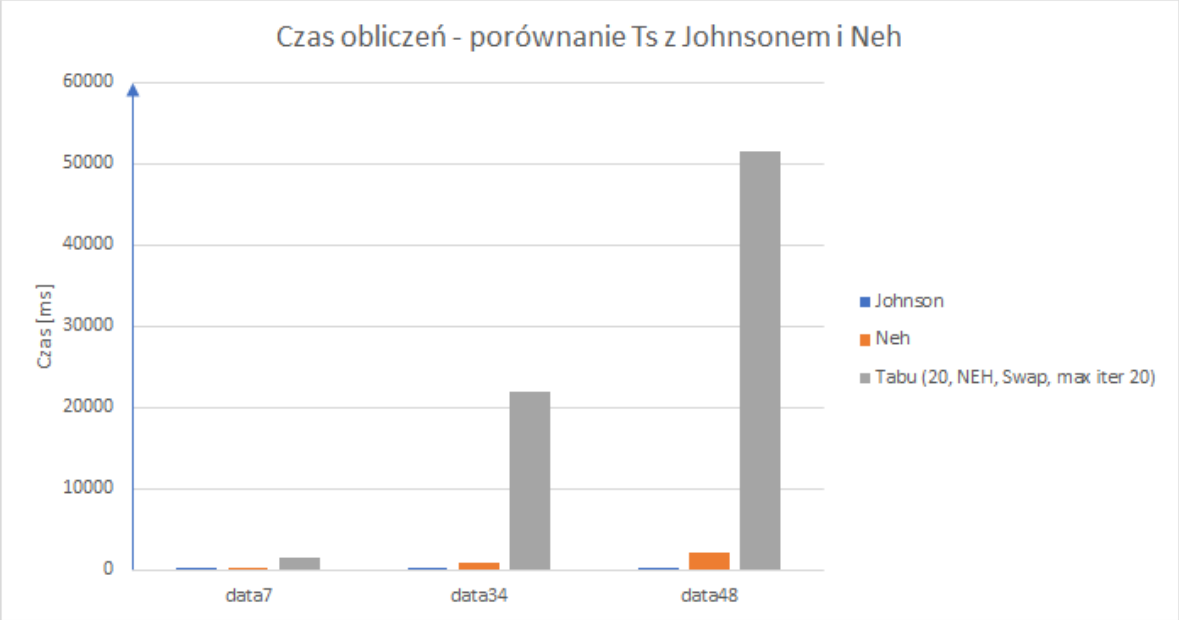
	C_max			Czas liczenia [ms]		
nr pliku	data7	data34	data48	data7	data34	data48
Tabu (20, NEH, Swap, max iter 20)	1251	2777	3089	1399,1	21850,9	51603,1
Johnson	1500	2958	3579	0,3	1,1	1,3
Neh	1278	2782	3123	57,9	797,7	2030,6
Tabu (2, NEH, Swap, max iter 20)	1251	2782	3114	1452,1	22242,8	51768,4
Tabu (200, NEH, Swap, max iter 20)	1251	2762	3089	1430,6	22275,9	50481,3
Tabu (20, NEH, Insert, max iter 20)	1278	2782	3123	1400,1	22322,5	52527,5
Tabu (20, NEH, Inverse, max iter 20)	1278	2782	3123	1387,8	21735,9	52382,0
Tabu (20, NEH, Swap, max time 60)	1251	2777	3089	60089,9	61541,2	63460,0
Tabu (20, NEH, Swap, without progress 20)	1251	2777	3089	1531,6	26947,7	92631,8
Tabu (20, NEH, Swap, max iter 50)	1251	2777	3089	3294,1	52581,8	124422,9
Tabu (20, Copy, Swap, without progress 20)	1251	2782	3126	2525,1	44694,5	101810,9
Tabu (20, Johnson, Swap, without progress20)	1259	2777	3107	1870,5	28712,4	104240,5
Tabu (20, NEH1, Swap, without progress 20)	1251	2753	3160	1689,8	32032,9	61308,9
Tabu (20, NEH2, Swap, without progress 20)	1259	2782	3098	1642,2	25217,2	108206,3
Tabu (20, NEH3, Swap, without progress 20)	1259	2782	3113	1637,7	27436,1	78372,4
Tabu (20, NEH4, Swap, without progress 20)	1521	2777	3089	2051,3	39318,7	126546,6

Rysunek 1: Tabela z wynikami pomiarów

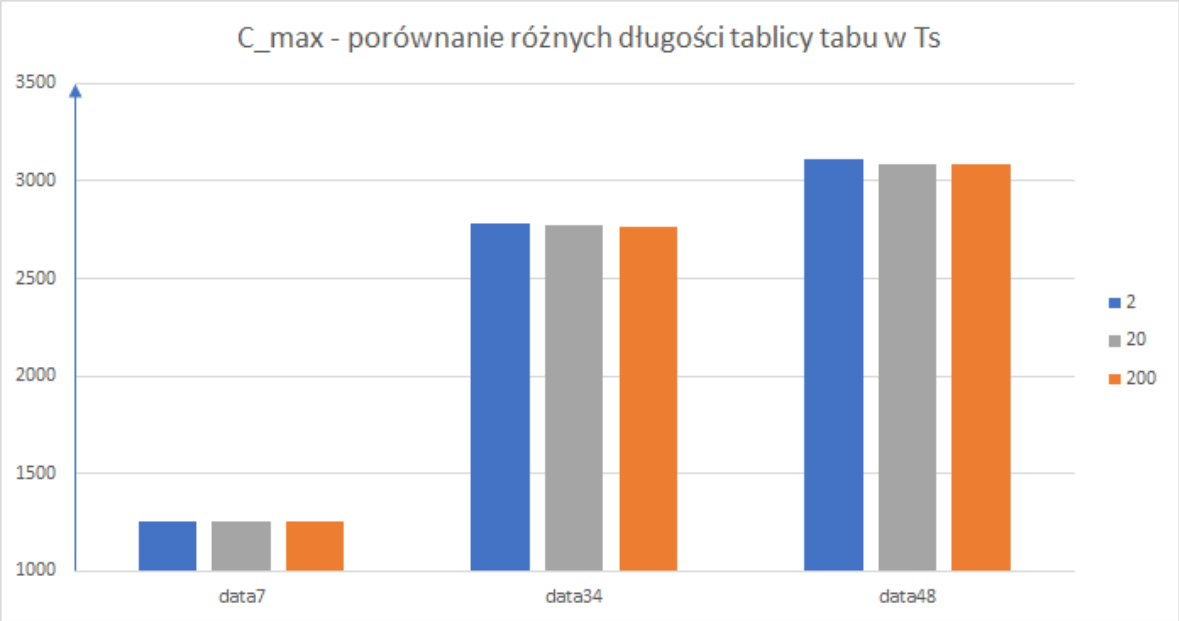
2 Wykresy



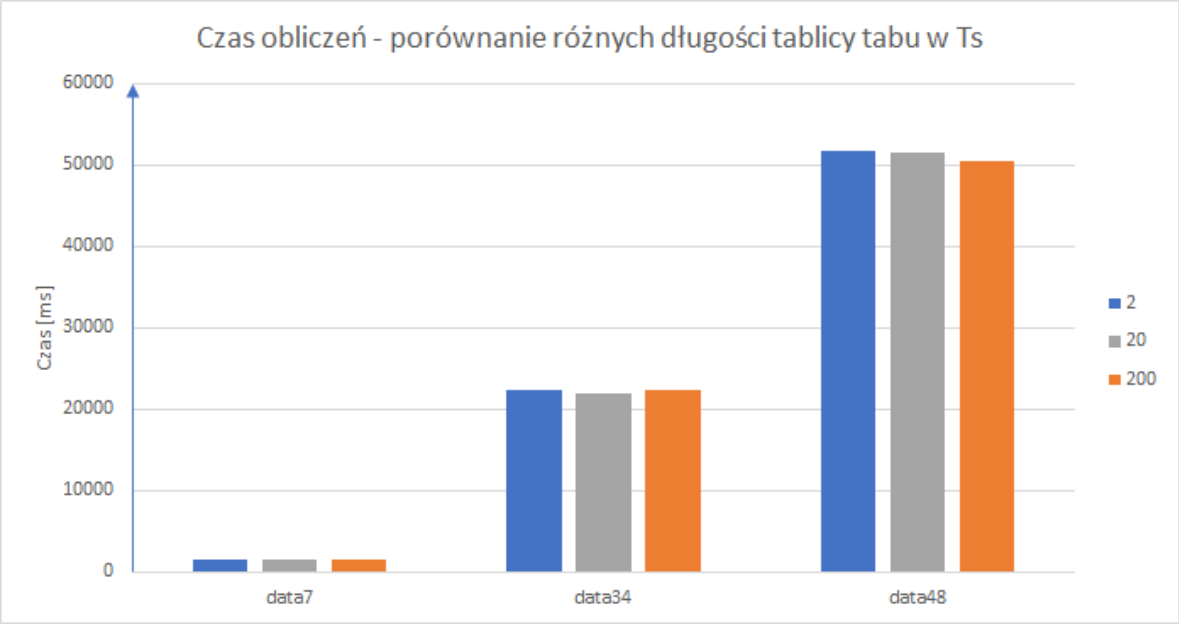
Rysunek 2: C_{max}



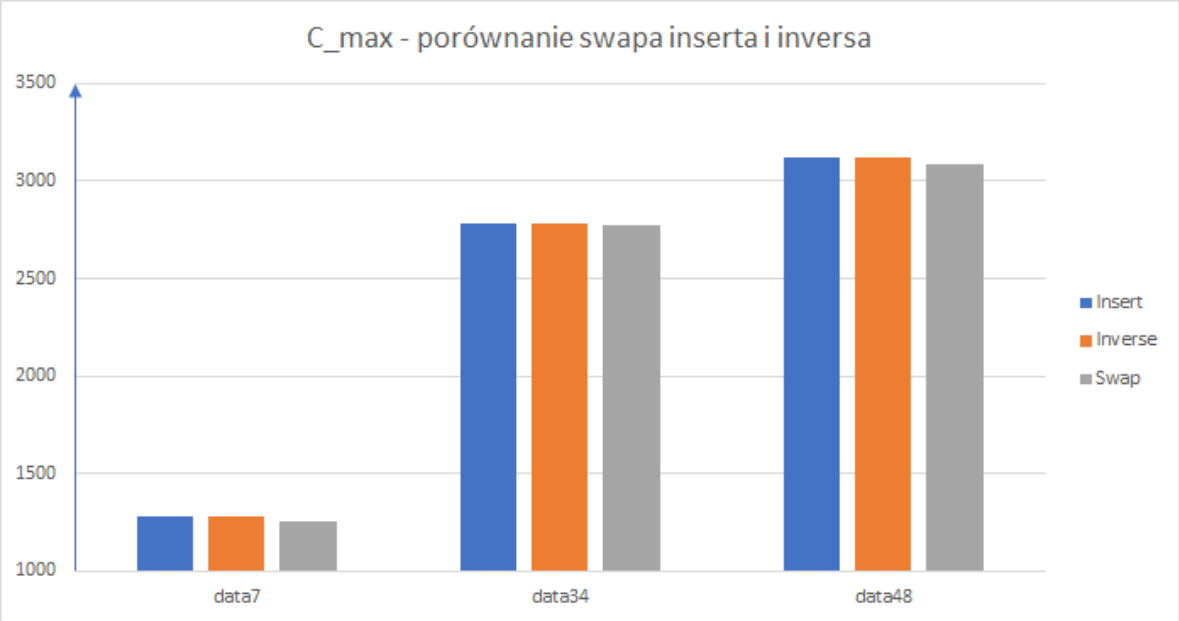
Rysunek 3: Czas obliczeń



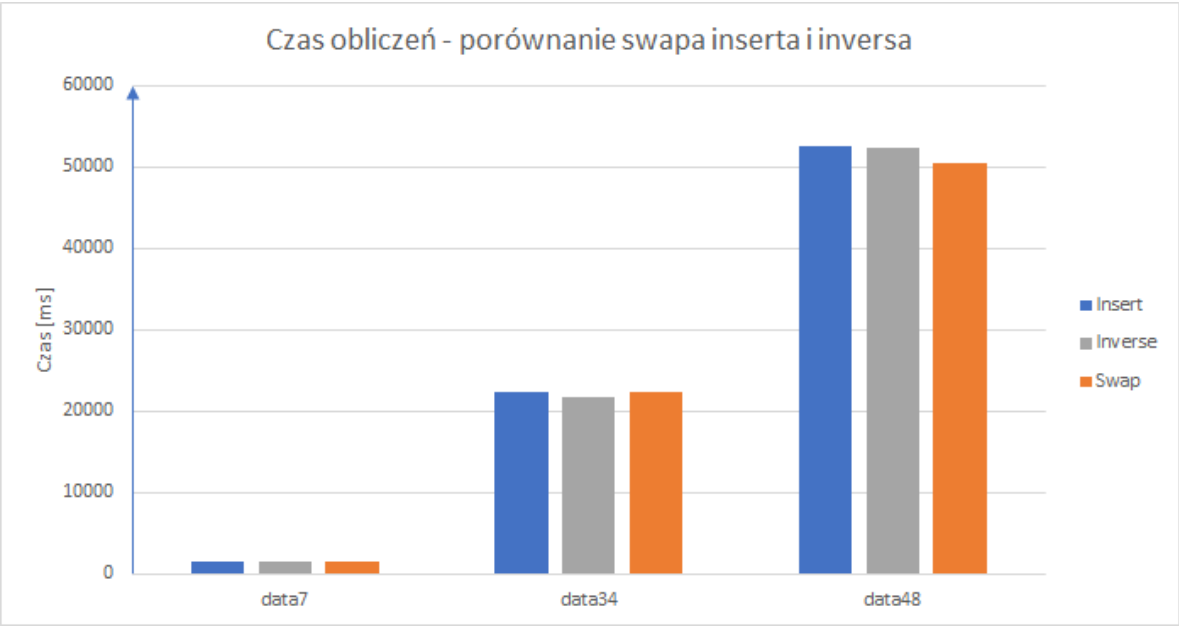
Rysunek 4: C_{max}



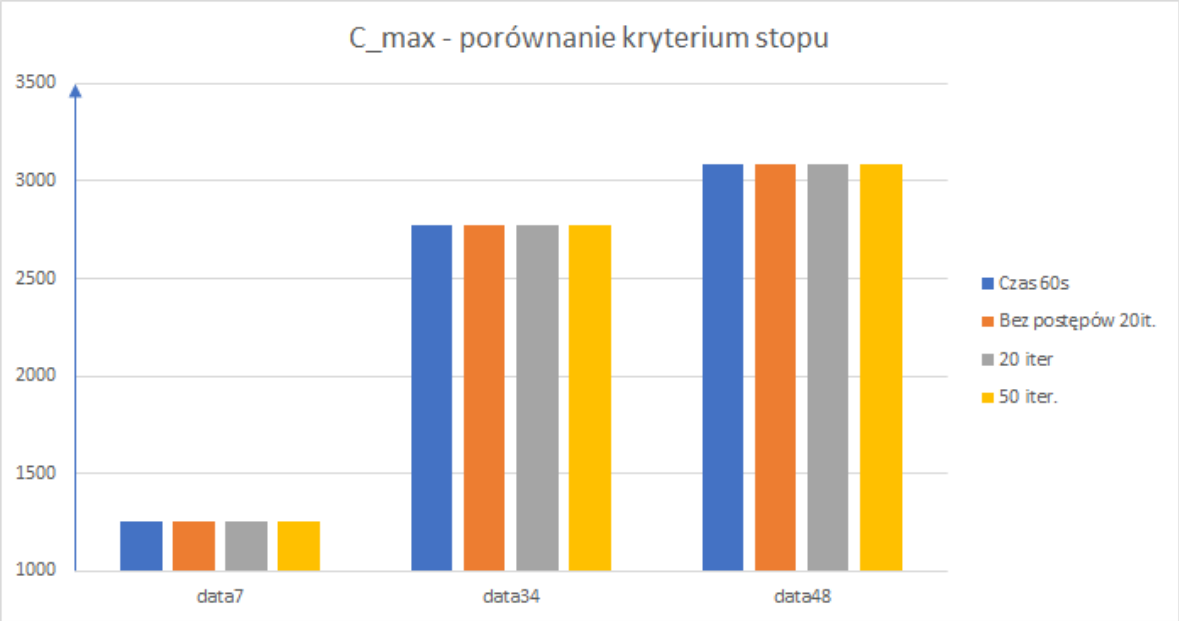
Rysunek 5: Czas obliczeń



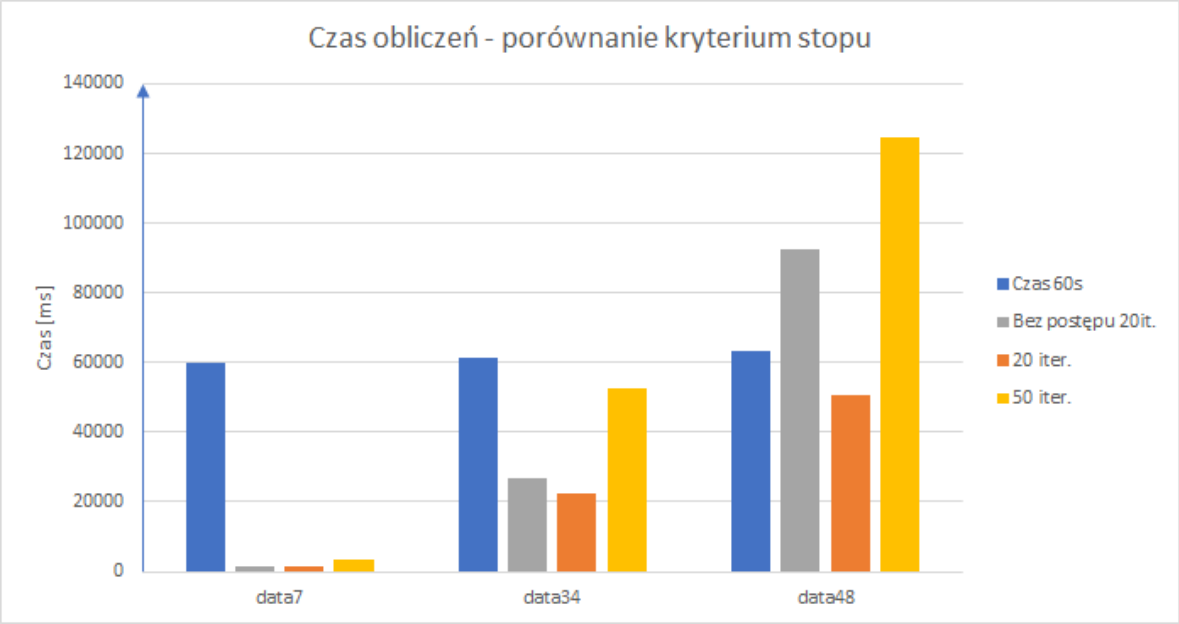
Rysunek 6: C_{max}



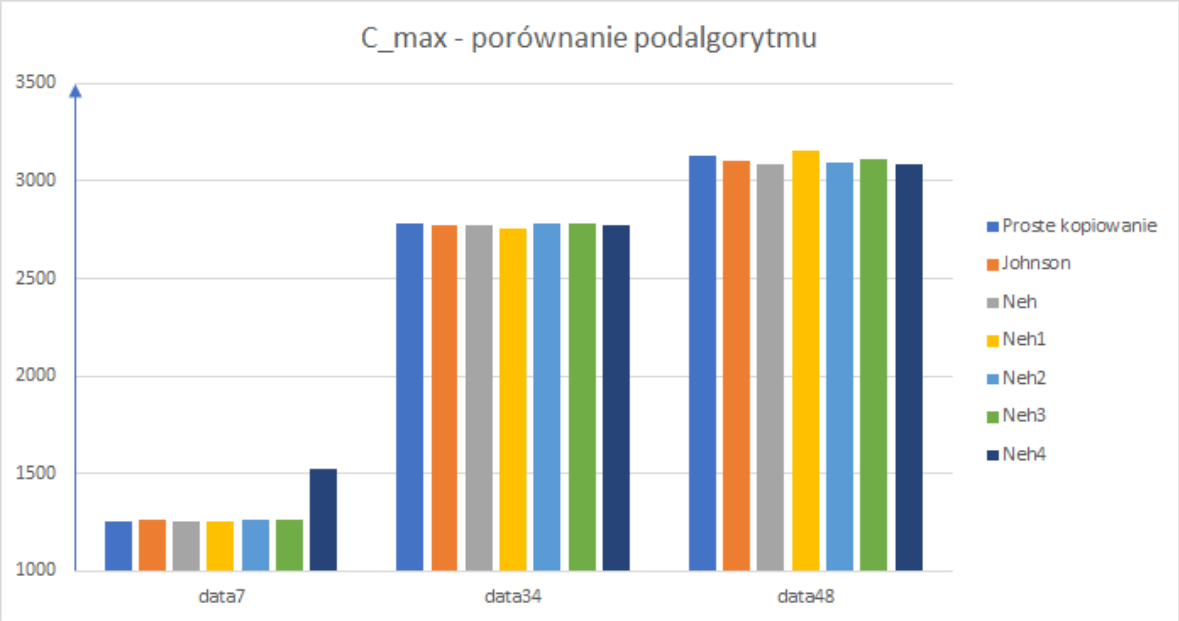
Rysunek 7: Czas obliczeń



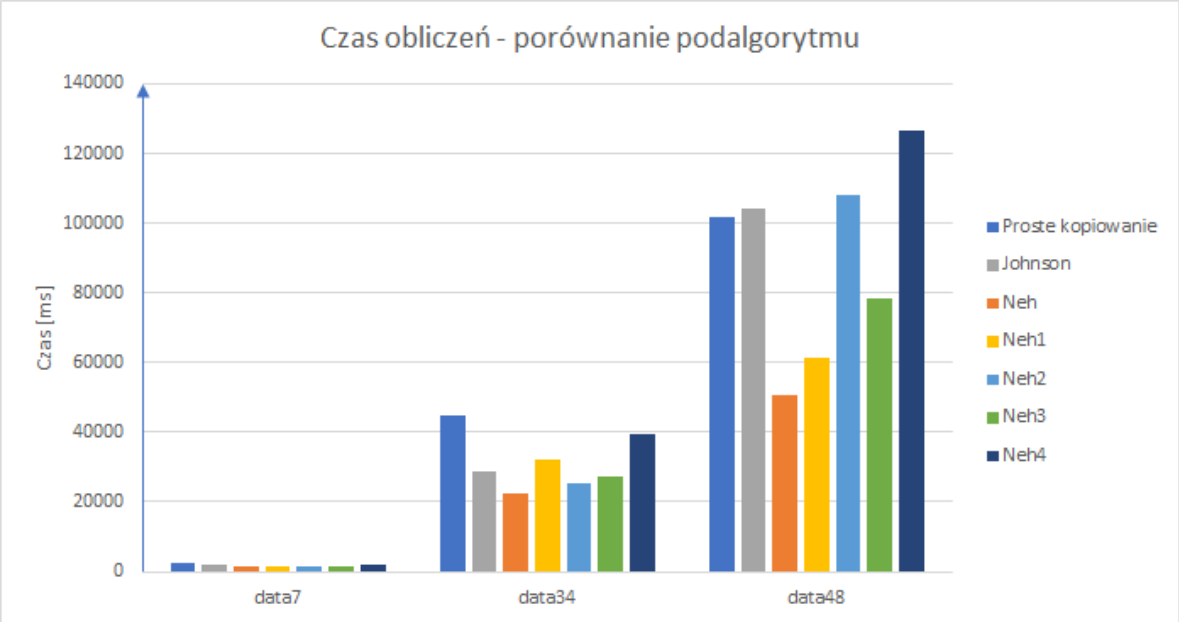
Rysunek 8: C_{max}



Rysunek 9: Czas obliczeń



Rysunek 10: C_{max}



Rysunek 11: Czas obliczeń

3 Wnioski

Jak pokazują wykresy 2 oraz 3 algorytm Tabu Search jest zdecydowanie wolniejszy od metody Johnsona lub algorytmu Neh. Wynika to najprawdopodobniej z tego, że przeprowadza dużą liczbę iteracji takich właśnie podalgorytmów. W zamian za to potrafi uzyskać lepsze rozwiązanie, choć oczywiście dalej nie daje gwarancji rozwiązania optymalnego. Większa długość tablicy tabu wpływa pozytywnie na wynik C_{max}, jednak nieznacznie (rys. 4). Czas obliczeń w zależności od rodzaju danych może się zarówno skrócić, jak i wydłużyć (rys. 5).

Z wykresów 6 i 7 wynika, że `swap` jest zazwyczaj szybszą i lepszą pod względem C_{max} metodą budowania tablicy sąsiedztwa. Wynika to z tego, że metody `insert` oraz `invert` są bardziej złożone obliczeniowo. Te metody mogą za to okazać się szybsze dla odpowiednich danych (plik `data.034` - `inverse`).

Kryterium stopu, przy dobranych parametrach, jak na rysunku 8 ma niewielki wpływ na wartość C_max, jednak już bardzo duży na czas wykonywania (rys 9). W praktyce dla danych niedużej objętości najlepiej sprawdzi się kryterium ilości iteracji bez żadnej poprawy wartości C_max - często trudno jest samodzielnie dobrać ilość iteracji tak, by trafić w punkt, po którym nowe iteracje nie przynoszą pożądanego efektu. Dla danych dużej objętości najlepiej sprawdzi się ograniczenie czasowe - daje ono gwarancję, że problem zostanie rozwiązany w sensownym czasie, jednak rozwiązanie to może być dalekie od optymalnego. Najprawdopodobniej najlepszy efekt dałoby połączenie kryterium czasowego oraz ilości iteracji bez poprawy rozwiązania.

Rysunki 10 oraz 11 pokazują wpływ doboru podalgorytmu - metody budowania rozwiązania. Można z nich wywnioskować, że pod względem czasowym i rezultatowym najlepiej poradziła sobie metoda Neh w podstawowej wersji.