Porównanie czasów wykonywania się na procesorze algorytmu NEH oraz symulowanego wyżarzania. Do wykonania tego posłużyły instancje wykorzystywane we wcześniejszej implementacji, a do zmierzenia czasu wykorzystana została funkcja time.time_ns() znajdująca się w bibliotece time. Początek zliczania umieszczamy przed samym rozpoczęciem algorytmu, a koniec umieszczamy przed wypisaniem kolejności po sortowaniu. Z racji tego, iż symulowane wyżarzanie działa bardzo losowo dla jednej instancji testowałem go trzykrotnie a w tabeli porównawczej umieściłem wynik z najmniejszym Cmaxem.

Nazwa instancji	NEH (Czas)	Symulowane wyżarzanie (Czas)
ta000	0,99 ms	2,99 ms
ta010	14,99 ms	15,99 ms
ta020	24,97 ms	29,98 ms
ta030	51,96 ms	55,96 ms
ta040	221,35 ms	36,97 ms
ta050	355,79 ms	68,97 ms
ta060	645,62 ms	134,92 ms
ta070	1659,04 ms	69,95 ms
ta080	2821,38 ms	134,91 ms
ta090	5035,11 ms	266,84 ms
ta100	21920,41 ms	267,84 ms

Nazwa instancji	NEH (Cmax)	Symulowane wyżarzanie (Cmax)
ta000	32	32
ta010	1127	1228
ta020	1656	1759
ta030	2257	2423
ta040	2801	2963
ta050	3267	3413
ta060	4036	4422
ta070	5336	5577
ta080	5937	6381
ta090	6680	7531
ta100	10808	11514

Jak można zauważyć dla początkowych instancji algorytm NEH miał przewagę nad symulowanym wyżarzaniem nie tylko w kwestii wykonywania się na procesorze, ale również w najbardziej optymalnym Cmax. Zwiększając stopniowo ilość zadań oraz maszyn poprzez wczytywanie instancji o coraz większym numerze możemy zauważyć jak symulowane wyżarzanie wykonuje się bardzo szybko w porównaniu do NEHa. Swoją szybkość zawdzięcza on losowości, która w dużej mierze determinuje działanie algorytmu. Niestety skutkuje to wybieraniem mniej optymalnej kolejności szeregowania, którą możemy zauważyć w tabeli porównawczej Cmax.

Badania dotyczące współczynnika wychładzania:

a) $\mu = 0.8$

Nazwa instancji	Symulowane wyżarzanie (Cmax)	Symulowane wyżarzanie (Czas w ms)
ta000	32	0
ta010	1197	9,99
ta020	1777	31,34
ta030	2377	31,26
ta040	2909	31,24
ta050	3476	46,85
ta060	4446	78,11
ta070	5683	46,87
ta080	6469	78,11
ta090	7620	171,88
ta100	11750	171,86

b) $\mu = 0.9$

Nazwa instancji	Symulowane wyżarzanie (Cmax)	Symulowane wyżarzanie (Czas w ms)
ta000	32	5,96
ta010	1195	21,98
ta020	1700	39,99
ta030	2406	74,97
ta040	2881	47,99
ta050	3441	91,96
ta060	4337	201,88
ta070	5506	98,34
ta080	6331	212,88
ta090	7279	368,78
ta100	11556	648,63

c) $\mu = 0.99$

Nazwa instancji	Symulowane wyżarzanie (Cmax)	Symulowane wyżarzanie (Czas w ms)
ta000	32	61,94
ta010	1127	494,71
ta020	1643	415,78
ta030	2263	1002,42
ta040	2830	763,56
ta050	3225	965,46
ta060	4140	1891,93
ta070	5451	994,45
ta080	6019	1885,90
ta090	6997	3710,87
ta100	11031	3729,86

Jak można zauważyć na powyższych tabelkach zwiększając współczynnik wychładzania poprawiamy wybór najbardziej optymalnego rozwiązania przez nasz algorytm, co skutkuje coraz mniejszymi wartościami Cmax. Jedynym minusem tego działania jest zwiększający się czas pracy algorytmu na procesorze.

Zwiększanie temperatury początkowej To powoduje poprawienie działania algorytmu pod względem zwracanego Cmax. Nie jest to jakaś znacząca poprawa, na przykład dla instancji ta090 algorytm ze 100 krotnie większą temperaturą początkową zwraca Cmax mniejsze o zaledwie 50. Znaczącą różnicą natomiast możemy zauważyć w czasie wykonywania się na procesorze, algorytm wykonuje się średnio 2 razy dłużej. Korzystniejsze efekty przynosi ustawienie temperatury końcowej na bardzo małą na przykład 1e-10, różnica zauważalna w zwracanym Cmax dla instancji ta090 wynosiła 150.