Optymalizacja Kombinatoryczna - Projekt 2

Wykonali:
Filip Rosiak - 151799 - filip.rosiak@student.put.poznan.pl
Jakub Stefański - 151876 - jakub.stefanski.1@student.put.poznan.pl

I. Informacje podstawowe

Temat projektu: Problem III - Capacitated Vehicle Routing Problem with Time Windows (problem marszrutyzacji z ograniczeniami pojemności i oknami czasowymi)

Opis problemu: Problem optymalizacyjny polegający na wyznaczeniu optymalnych tras przewozowych dla pewnej ściśle określonej liczby środków transportu, której zadaniem jest obsłużenie zbioru klientów znajdujących się w różnych punktach przy zachowaniu ograniczeń. Kryterium optymalizacji jest całkowity koszt transportu (wyrażony odległościowo, cenowo lub czasowo).

Kod źródłowy - język C++.
Środowisko - Linux.

Obsługa programu:

- sugerowana kompilacja: g++ -std=c++11 rosiakstefanskicvrptw2.cpp
- uruchamianie programu: ./a.out <plik_wejściowy> <plik_wyjściowy> <ilość klientów do wczytania(-1 aby wczytać wszystkich klientów)>

II. Rozwiązania problemu

Algorytm opiera się na metodzie *tabu search* polegającej na przeszukiwaniu rozwiązań pokrewnych do wygenerowanego wcześniej rozwiązania bazowego (w naszym przypadku posłużyliśmy się algorytmem zachłannym z pierwszego projektu) z wykluczeniem pewnej ilości ostatnio znalezionych polepszonych rozwiązań. Stanowią one listę tabu, do których nie można się wrócić, co zapobiega krążeniu wokół lokalnych optimów funkcji celu. https://pl.wikipedia.org/wiki/Przeszukiwanie_tabu

Na początku program znajduje zachłannie startowe rozwiązanie problemu, dla którego później generowane jest sąsiedztwo o podanym rozmiarze. Przy tworzeniu sąsiada tworzona jest kopia rozwiązania startowego, W pierwszym kroku modyfikacji trasy algorytm szuka możliwej wymiany pary klientów między dwoma losowo wybranymi

trasami. Następnie trasy, na których dokonywaliśmy wymiany poddawane są algorytmowi *two-opt*, polegającemu na powtarzaniu procedury odwrócenia pewnego wewnętrznego fragmentu trasy, do momentu znalezienia nowego najlepszego (i możliwego) ułożenia klientów. Po zakończonym „mieszaniu" nowa trasa dodawana jest do listy sąsiadów. Następnie algorytm wybiera najlepszego sąsiada, jednocześnie sprawdzając czy nie figuruje on już w tabu, ustanawia go aktualnie najlepszym rozwiązaniem oraz dodaje do listy tabu, teraz na jego podstawie będą generowane nowe potencjalne rozwiązania. Cały algorytm wykonuje się w pętli do momentu osiągnięcia ustalonego limitu czasowego.

Krótki opis funkcji występujących w programie:

droga_mozliwa() – sprawdza czy podana trasa może istnieć

two_opt_swap() – implementacja przekształcenia *two-opt* – odwrócenie fragmentu wewnątrz trasy

koszt_trasy() – zlicza koszt podanej trasy

two_opt() – szukanie najlepszej modyfikacji trasy metodą *two-opt*

wymien_klientow() – wymienia klientów między dwoma podanymi trasami

sumaryczny_czas_tras() – zlicza koszt tras całego podanego rozwiązania

czy_w_tabu() – sprawdza czy podane rozwiązanie jest w liście tabu

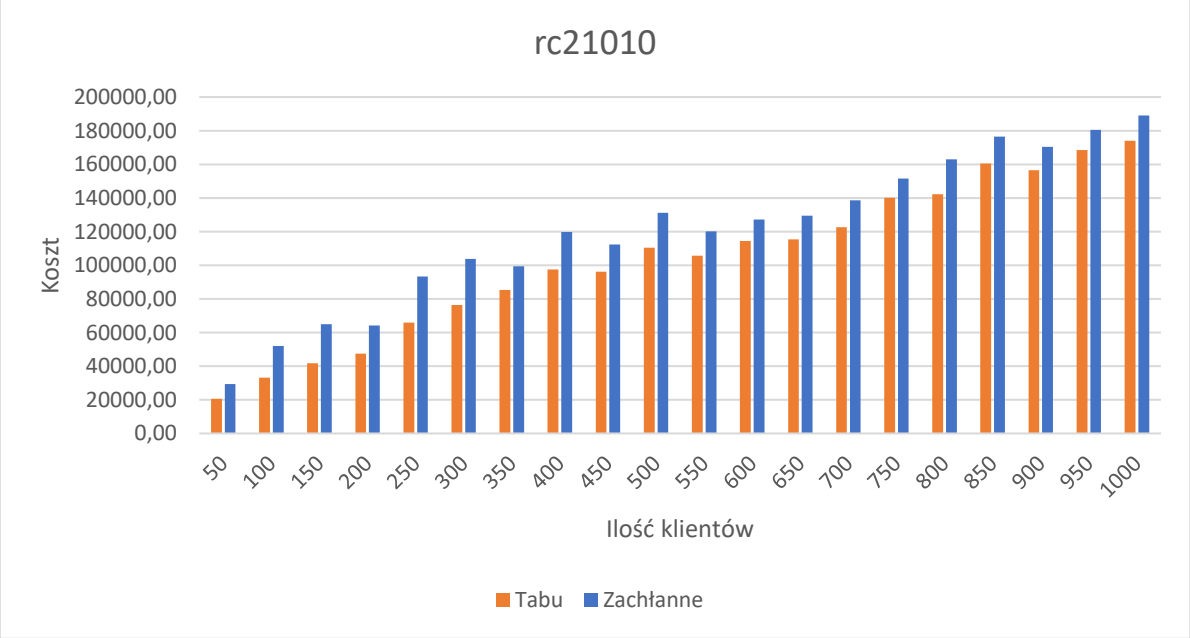
generuj_sasiedztwo() – generuje zestaw nowych pozmienianych rozwiązań („sąsiadów") na bazie podanego

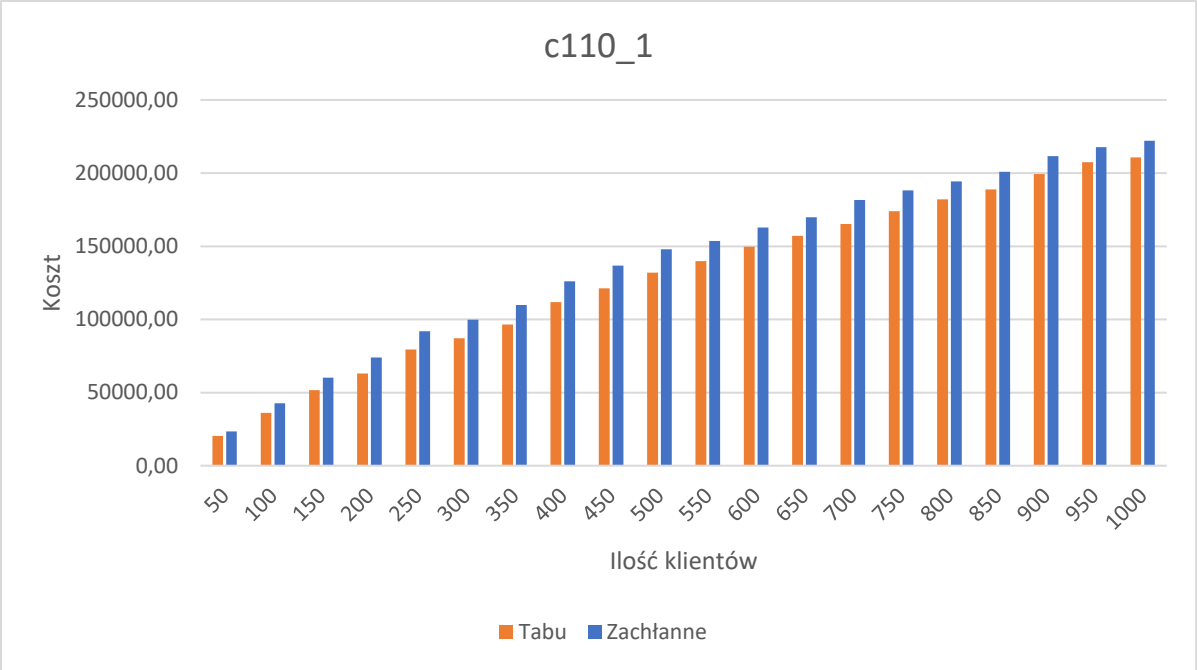
tabu() – implementacji algorytmu tabu

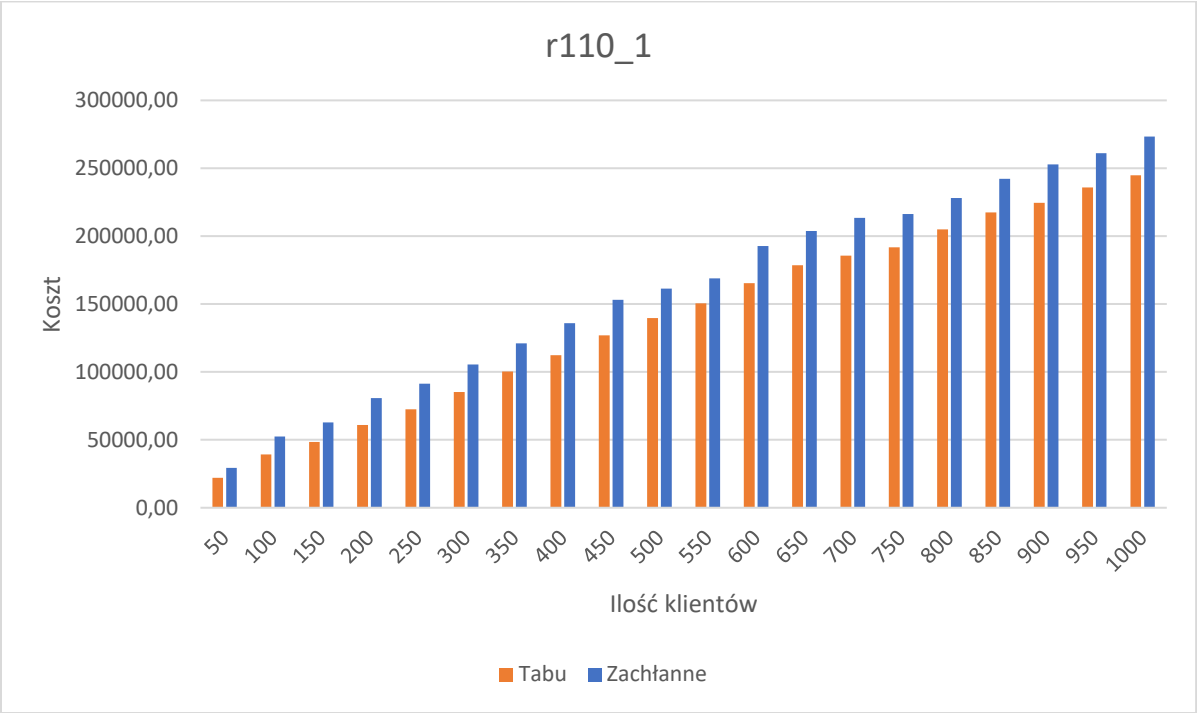
III. Badania efektywnościowe dla instancji rc21010, c110_1, r110_1

	rc21010		c110_1		r110_1	
	Zachłanny	Tabu	Zachłanny	Tabu	Zachłanny	Tabu
50	29271,59	20550,31	23449,00	20320,53	29330,81	22109,21
100	52033,74	33106,87	42719,48	36057,46	52441,45	39205,20
150	64946,70	41698,44	60127,41	51642,55	62946,49	48555,48
200	64109,84	47381,53	73920,87	63039,07	80663,24	60941,86
250	93397,55	65927,42	91856,72	79411,44	91456,40	72589,42
300	103681,89	76307,19	99856,15	87150,36	105432,63	85318,49
350	99483,27	85287,76	109758,12	96590,69	121113,71	100247,36
400	119785,34	97405,81	126031,40	111784,71	135886,47	112427,50
450	112385,59	96142,42	136859,13	121117,62	153201,52	127018,68
500	131159,19	110434,08	147826,69	131922,35	161430,19	139584,79
550	120125,42	105664,79	153620,36	139775,35	168974,16	150570,45
600	127094,07	114417,58	162864,73	149613,39	192814,93	165430,51
650	129500,93	115384,34	169783,40	157199,26	203734,66	178647,19
700	138570,95	122686,50	181529,74	165270,14	213583,43	185586,05
750	151641,57	140223,67	188245,30	173911,40	216428,61	191897,71
800	162989,71	142225,39	194285,27	182092,83	228176,34	205108,85
850	176451,86	160453,65	200882,32	188762,53	242218,68	217565,80
900	170305,83	156457,01	211542,45	199390,91	252965,39	224621,34
950	180430,48	168427,86	217758,14	207444,55	261148,71	235891,22
1000	188953,68	173992,97	222111,28	210671,84	273473,30	244739,01

Dla każdego testu algorytm tabu wykonywał się 5 minut.







IV. Wnioski

Z powyższych danych wynika, że algorytm tabu znalazł lepsze rozwiązania od początkowych. W przypadku instancji rc21010 średnia poprawa w procentach wyniosła 17,17%, w przypadku instancji c110_1 - 9,93%, a dla instancji r110_1 - 15,78%.