Projektowanie efektywnych algorytmów - Projekt 2

Problem komiwojażera - Tabu Search.

.

Termin zajęć: Czwartek 09:15-11:00
Prowadzący: dr inż. Dariusz Banasiak
Adam Filipowicz 221713

1 Wstęp

W projecie implementowałem algorytm Tabu Search dla problemu komiwojażera. Stworzony algorytm uwzględnia mechanizm dywersyfikacji przeszukiwania przestrzeni rozwiązań i sąsiedztwo typu zamiany dwóch miast. W projekcie najpierw testowałem uśrednione wyniki dla 4 zmieniających się parametrów tworząc w ten sposób 4-wymiarową hiperkostkę rozwiązań, a następnie analizując jej pojedyńcze części wyciągałem wnioski na temat najlepszych wartości dla poszczególnych rozwiarów instancji. Następnie porównałem wartości zwrócone oraz czas wykonania programu dla realnych problemów pobranych ze strony 'comopt.ifi.uni-heidelberg.de/software/TSPLIB95/' – z wartościami i czasem z projektu poprzedniego (programowania dynamicznego) – również dla różnych parametrów, tym razem jednak z pomocą wcześniej wyrobionej intuicji. Grafu używałem do reprezentacji odległości między miastami. Do reprezentacji grafu użyłem macierzy, w której przechowywane są wartości poszczególnych krawędzi, np. wartość M[i][j] to wartość krawędzi między wierzchołkami (i j). Zużycie pamięci dla grafu to O(V^2), gdzie V to ilość wierzchołków. Dodanie, sprawdzenie istnienia i usunięcie krawędzi mają złożoność O(1). Sprawdzenie stopnia wierzchołka ma złożoność O(V).

Wynik działania algorytmu przechowuję w tablicy o długości V w postaci znalezionej permutacji oraz w zmiennej przechowującej długosc całkowitą znalezionej ścieżki.

1.1 Problem komiwojażera

Problem komiwojażera polega na znalezieniu najkrótszego cyklu w grafie pełnym, gdzie każdy wierzchołek odwiedzamy dokładnie raz.

Algorytm Tabu Search

Zaimplementowany algorytm to algorytm Tabu Search. Wykorzystuje on przeszukiwanie lokalne pod kątem zdefiniowanego sasiedztwa. Polega na dynamicznej zmianie tego sąsiedztwa w zależności on informacji zebranych w poprzednich momentach działania algorytmu. Informacje jednak ulegają zapomnieniu po pewnym zdefiniowanym czasie. Dodatkowo, jeśli po pewnej ilości iteracji nie nastąpi poprawa znalezionego rozwiązania następuje wylosowanie zupełnie nowego i wyczyszczenie wszystkich zebranych do tej pory informacji.

Zatem mamy zdefiniowane zmienne jako:

- ilość iteracji – kryterium zakończenia algorytmu, czyli ilość prób znalezienia rozwiązania w sąsiedztwie
- lista tabu – lista ruchów zakazanych, czyli ostatnio używanych zamian wierzchołków
- kadencja – długość przetrzymywania danej pary wierzchołków w liście tabu
- kryterium aspiracji – na jakiej podstawie możemy jednak przejrzeć listę tabu w poszukiwaniu rozwiązań
- zdarzenie krytyczne – 'reset' algorytmu i listy tabu w przypadku braku poprawy rozwiązania

Implementacja

Początkową permutację tworzę losując wierzchołek do zamiany dla każdego wierzchołka oprócz pierwszego zaczynając od ostatniego. Następnie obliczam początkową ścieżkę z wylosowanej permutacji. Dla podanej ilości iteracji wykonuję:

- poszukiwanie najbardziej optymalnej trasy dla danej iteracji dla wszystkich sąsiadów z kryterium sąsiedztwa pod warunkiem że dwa wierzchołki do zamiany nie występują na liście tabu
- jeśli po przejrzeniu wszystkich sasiadów wynik nie został poprawiony, to dla zadanej wartości aspiracji sprawdzam taką liczbę rozwiązań z listy tabu
- zmniejszam kadencję wszystkich elementów na liście tabu i usuwam pozycje na których kadencja jest równa 0
- dodaję parę wierzchołków znalezioną w obecnej iteracji do listy tabu
- jeśli rozwiązanie znalezione w obecnej iteracji jest lepsze od dotychczasowego, to uaktualniam dotychczasowe rozwiązanie. W przeciwnym wypadku zwiększam licznik do zdarzenia krytycznego

- jeśli licznik zdarzenie krytycznego osiągnął zadany pułap: generuję nową losową permutację, zeruję listę tabu oraz licznik zdarzenia krytycznego

Listę tabu przechowuję w tablicy wielkości podanej kadencji przyjmującą struktury z trzema integerami: wierzchołku o mniejszym numerze, wierzchołku o większym numerze oraz kadencji dla danej pary wierzchołków.

Przykład algorytmu dla 4 wierzchołków

Załóżmy że problem jest asymetryczny opisany poniższą macierzą:

Miasta	1	2	3	4
1	0	41	130	64
2	24	0	58	32
3	13	52	0	20
4	82	40	14	0

Załóżmy, że ilość iteracji wynosi 3 (I – licznik iteracji), kadencja(K) wynosi 2, licznik zdarzenia krytycznego wynosi 2 (ZK – obecny licznik), a aspiracja(A) wynosi 1.

Załóżmy także, że wylosowana permutacja miast to 2-1-3-4, zatem obecna długość ścieżki to: $24 + 130 + 20 + 40 = 214$.

W poniższej tabeli ujałem wszystkie możliwości dla wszystkich kolejnych iteracji oraz bieżącą listę tabu w postaci: {mniejszy wierzchołek-wiekszy wierzchołek, kadencja;...}:

I	Lista tabu	ZK	Permutacja	Zamienione wierzchołki	Długość ścieżki
1	$\emptyset$	0	2-1-3-4	2-1	201
				2-3	100
				2-4	296
				1-3	175
				1-4	100
				3-4	154
2	{2-3, 2}	0	3-1-2-4	3-1	296
				3-2	Zabronione
				3-4	201
				1-2	154
				1-4	214
				2-4	175
3	{1-2, 2; 2-3, 1}	1	3-2-1-4	3-2	175(aspiracja)
				3-1	201
				3-4	214
				2-1	Zabronione
				2-4	201
				1-4	296

Po trzech iteracjach okazało się, że rozwiązanie algorytmu nastąpiło w iteracji pierwszej. Minimalna znaleziona długość ścieżki wynosi 100, a znaleziona permutacja to 3-1-2-4. Gdyby była jeszcze jedna iteracja nastąpiłoby zdarzenie krytyczne (ponieważ rozwiązanie nie poprawiło się od dwóch iteracji) i zostałaby wylosowana nowa permutacja.

W iteracji 2 nie zostało użyte kryterium aspiracji, ponieważ z listy tabu nie może być ściągnięta właśnie dodana para wierzchołków. W iteracji 3 zostało już użyte kryterium aspiracji.

2 Plan eksperymentu

- Struktury danych są alokowane dynamicznie.
- Program testowany był dla znanych wyników dla ilości miast równej 17, 21, 24, 20, 48 oraz 120 dla grafu symetrycznego oraz dla ilości miast równej 17, 43, 53, 70 i 124 dla grafu asymetrycznego.
- Każdy pomiar losowy powtórzony jest 1000 razy za każdym razem dla tego samego grafu, a następnie liczona jest średnia. Wszystkie możliwości parametrów wynoszą: 500, 1000, 1500, 2000, 2500 iteracji; n, $2n$, $3n$ kadencji; $\frac{i}{100}$, $\frac{i}{50}$, $\frac{i}{25}$ licznika krytycznego oraz $\frac{n}{10}$, $\frac{n}{5}$ aspiracji, gdzie n to liczba wierzchołków w grafie, a i to liczba iteracji.
- Do pomiarów czasu używałem funkcji QueryPerformanceCounter. W pętli zapisywałem czasy dla kolejnych populacji i liczyłem średnią.
- Program napisany został w języku C++ w środowisku Microsoft Visual Studio 2017.
- Waga krawędzi, jest liczbą z przedziału (0,$RAND\_MAX$) wylosowaną za pomocą funkcji rand().
- Graf jest pełny zatem generuje wszystkie możliwe krawędzie (oprócz tych łączących wierzchołek z samym sobą). Graf może być symetryczny lub asymetryczny.

3 Zestawienie i analiza wyników

3.1 Analiza parametrów dla losowych grafów

Wyniki dla 10 miast

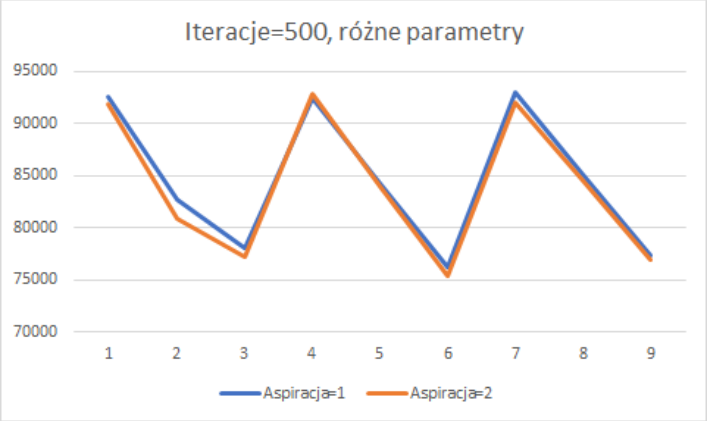
Iteracje	Aspiracja	Kadencja	Krytyczne	Średni wynik	Średni czas
500	1	10	5	92573	0.0000575453
	2			91896	0.0000580435
	1		10	82751	0.0000550297
	2			80875	0.0000512542
	1		20	78113	0.0000521376
	2			77230	0.0000522746
	1	20	5	92409	0.0000868754
	2			92864	0.0000896842
	1		10	84269	0.0000827352
	2			83965	0.0000862382
	1		20	76280	0.0000808257
	2			75378	0.0000793709
	1	30	5	92988	0.0000992736
	2			92044	0.000103171
	1		10	85126	0.000100601
	2			84607	0.000105412
	1		20	77413	0.000101715
	2			76946	0.000103238
1000	1	10	10	81682	0.0000980855
	2			80966	0.000103168
	1		20	77621	0.000101445
	2			76920	0.0000988503
	1		40	76924	0.000108103
	2			77068	0.00011837
	1	20	10	83531	0.000170262
	2			85238	0.000160306
	1		20	76740	0.000156788
	2			76468	0.000152797
	1		40	74508	0.000164947
	2			73741	0.000162841
	1	30	10	85207	0.000209151
	2			84925	0.00021643
	1		20	77246	0.000207738
	2			76657	0.000212499
	1		40	73203	0.000202394
	2			72933	0.00021064

Iteracje	Aspiracja	Kadencja	Krytyczne	Średni wynik	Średni czas
1500	1	10	15	79439	0.000163306
	2			77832	0.000157271
	1		30	77399	0.000158812
	2			76724	0.000186540
	1		60	76291	0.000181861
	2			76775	0.000189984
	1	20	15	79506	0.000256718
	2			78953	0.000260545
	1		30	74773	0.000245228
	2			73979	0.000244918
	1		60	73852	0.000250241
	2			73418	0.000250762
	1	30	15	79679	0.000302207
	2			79944	0.000298938
	1		30	74646	0.000268817
	2			74329	0.000276513
	1		60	72983	0.000275935
	2			72622	0.000287974
2000	1	10	20	78086	0.000197865
	2			77467	0.000204368
	1		40	76815	0.000222153
	2			77365	0.000222766
	1		80	76539	0.000241068
	2			76610	0.000268335
	1	20	20	76288	0.000318678
	2			76003	0.000317126
	1		40	74225	0.000314907
	2			74091	0.000311584
	1		80	73891	0.000323136
	2			73622	0.000339446
	1	30	20	77366	0.000398845
	2			76077	0.000398846
	1		40	73635	0.000381336
	2			73391	0.000385271
	1		80	72695	0.000398661
	2			73057	0.000414595

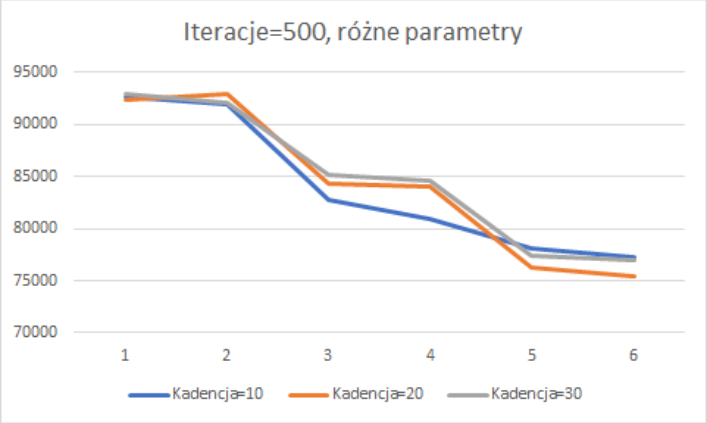
Iteracje	Aspiracja	Kadencja	Krytyczne	Średni wynik	Średni czas
2500	1	10	25	77796	0.000306368
	2			77503	0.000312701
	1		50	76778	0.000285612
	2			76846	0.000299243
	1		100	76638	0.000312812
	2			76709	0.000321956
	1	20	25	75883	0.000443444
	2			75076	0.000447667
	1		50	74078	0.000440941
	2			73934	0.000442246
	1		100	73565	0.000422701
	2			73746	0.000445269
	1	30	25	75355	0.000500425
	2			75288	0.000496181
	1		50	72722	0.000473096
	2			73133	0.000481191
	1		100	73146	0.00046875
	2			73097	0.00050489

Średni czas podany jest w sekundach.

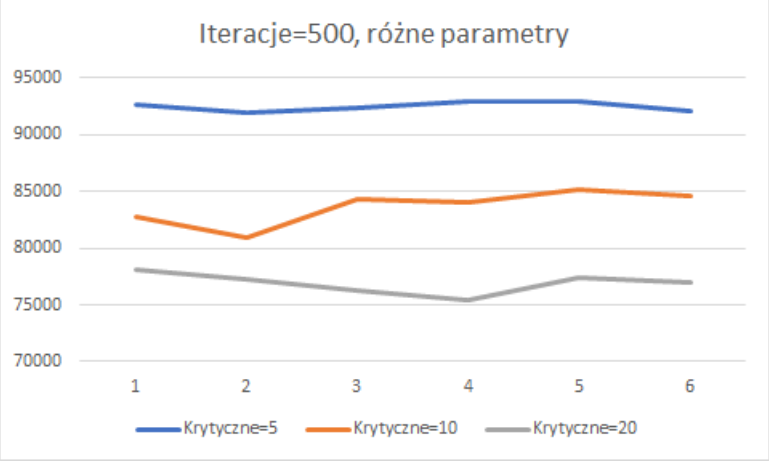
Najważniejsze wykresy



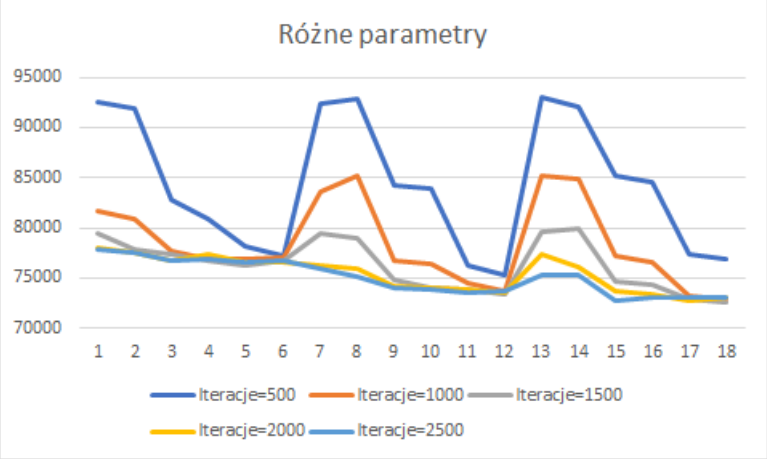
Rysunek 1: Porównanie dwóch różnych aspiracji dla liczby iteracji=500 i wszystkich innych parametrów różnych



Rysunek 2: Porównanie trzech różnych kadencji dla liczby iteracji=500 i wszystkich innych parametrów różnych



Rysunek 3: Porównanie trzech różnych licznikow zdarzen krytycznych dla liczby iteracji=500 i wszystkich innych parametrów różnych



Rysunek 4: Porównanie pięciu różnych iteracji dla wszystkich innych parametrów różnych

Wnioski

- liczba aspiracji wywołuje małą różnicę wyników zarówno dla liczby iteracji równej 500 jak i innych
- najbardziej optymalne wyniki występują najczęściej dla kadencji równej n czyli 10 zarówno dla liczby iteracji równej 500 jak i innych
- licznik krytyczny ma duże znaczenie na wynik i dla liczby równej 20 jest on najbardziej optymalny
- zwiększenie liczby iteracji ma wpływ na poprawę wyniku dla wszystkich innych parametrów. Warto jednak zauważyć, że wraz z liniowym wzrostem liczby iteracji – poprawa następuje coraz wolniej

Wyniki dla 50 miast

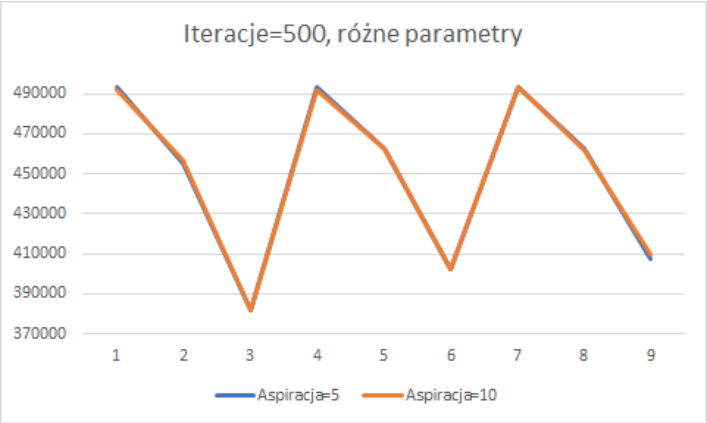
Iteracje	Aspiracja	Kadencja	Krytyczne	Średni wynik	Średni czas
500	5	50	5	493355	0.00769973
	10			492178	0.00758363
	5		10	454915	0.00762215
	10			456886	0.00768367
	5		20	381845	0.00772216
	10			381620	0.00779734
	5	100	5	493260	0.0100906
	10			491826	0.00960976
	5		10	462420	0.00983405
	10			462252	0.00986645
	5		20	402317	0.0101386
	10			402707	0.0100041
	5	150	5	493662	0.0118513
	10			493444	0.0117035
	5		10	462318	0.0118718
	10			461588	0.0117777
	5		20	407752	0.0119966
	10			409903	0.0119845
1000	5	50	10	445295	0.0158931
	10			440963	0.0150816
	5		20	373024	0.0154268
	10			372724	0.0150732
	5		40	320796	0.0154008
	10			323090	0.0153738
	5	100	10	448583	0.019356
	10			446916	0.0194794
	5		20	394078	0.0196063
	10			392592	0.0196937
	5		40	311483	0.0199587
	10			316229	0.0202313
	5	150	10	449853	0.0235493
	10			448069	0.0238162
	5		20	395103	0.0246039
	10			397068	0.024408
	5		40	316763	0.0245657
	10			322896	0.0245656

Iteracje	Aspiracja	Kadencja	Krytyczne	Średni wynik	Średni czas
1500	5	50	15	405018	0.0224875
	10			404270	0.0228831
	5		30	334721	0.0232077
	10			339059	0.0231849
	5		60	303187	0.0233219
	10			304966	0.0232698
	5	100	15	417913	0.029622
	10			422885	0.0296271
	5		30	341872	0.0308318
	10			342028	0.0309272
	5		60	285969	0.0307239
	10			289649	0.0307167
	5	150	15	420400	0.0364988
	10			420574	0.0364255
	5		30	356094	0.0362315
	10			351298	0.0363482
	5		60	283168	0.0380294
	10			285030	0.0370601
2000	5	50	20	370840	0.0297569
	10			375449	0.0298754
	5		40	321272	0.0314664
	10			325393	0.0319307
	5		80	298447	0.0306274
	10			301168	0.0299616
	5	100	20	390959	0.0386579
	10			393479	0.0385928
	5		40	310536	0.0392072
	10			318447	0.039259
	5		80	274846	0.0396015
	10			277507	0.0395455
	5	150	20	394601	0.0474544
	10			396471	0.0476585
	5		40	324087	0.0487162
	10			322652	0.0493196
	5		80	266016	0.0485845
	10			267874	0.0485247

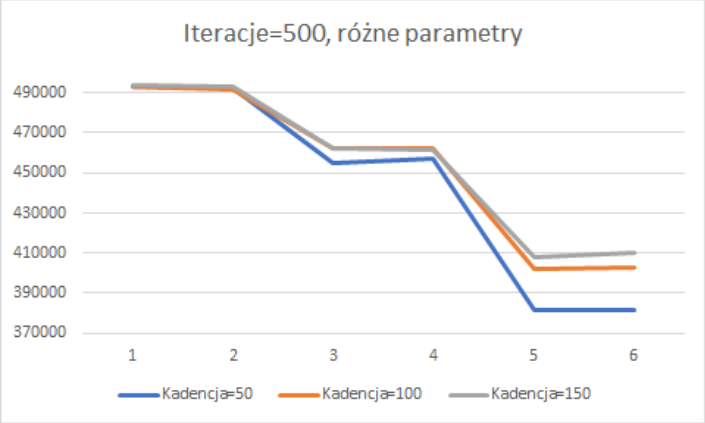
Iteracje	Aspiracja	Kadencja	Krytyczne	Średni wynik	Średni czas
2500	5	50	25	351313	0.0357979
	10			355028	0.0357178
	5		50	305957	0.0357561
	10			312327	0.0357476
	5		100	294993	0.0359229
	10			298706	0.0360085
	5	100	25	364820	0.0483334
	10			365868	0.0483851
	5		50	295802	0.0485541
	10			298366	0.0486707
	5		100	264357	0.0483069
	10			267112	0.0484287
	5	150	25	376697	0.0613319
	10			374935	0.0609386
	5		50	296962	0.0614107
	10			298830	0.0612733
	5		100	256511	0.0610205
	10			260330	0.0604511

Średni czas podany jest w sekundach.

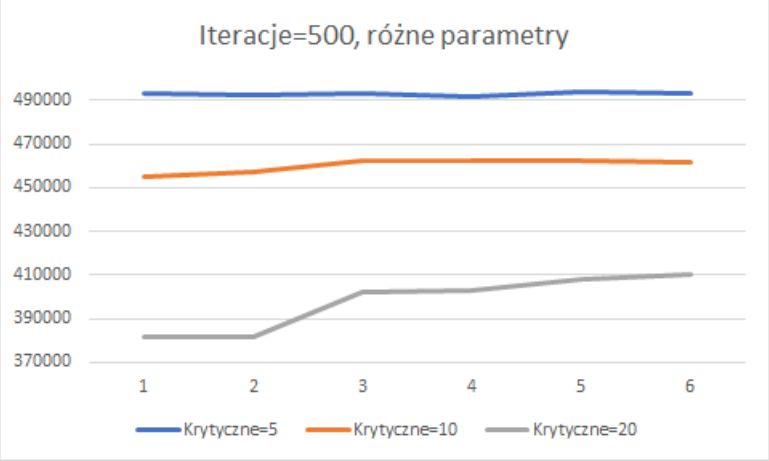
Najważniejsze wykresy



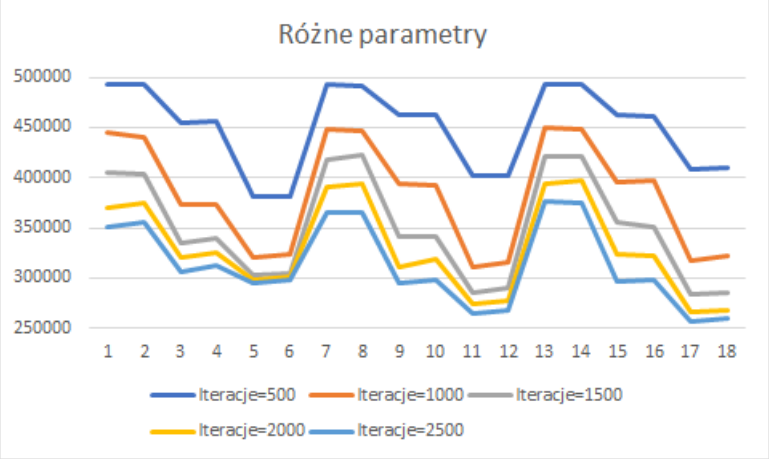
Rysunek 5: Porównanie dwóch różnych aspiracji dla liczby iteracji=500 i wszystkich innych parametrów różnych



Rysunek 6: Porównanie trzech różnych kadencji dla liczby iteracji=500 i wszystkich innych parametrów różnych



Rysunek 7: Porównanie trzech różnych licznikow zdarzen krytycznych dla liczby iteracji=500 i wszystkich innych parametrów różnych



Rysunek 8: Porównanie pięciu różnych iteracji dla wszystkich innych parametrów różnych

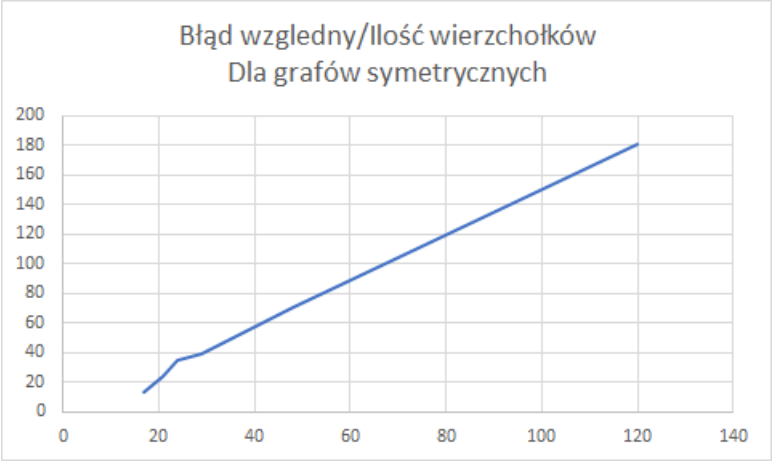
Wnioski

- liczba aspiracji wywołuje niezauważalną różnicę wyników zarówno dla liczby iteracji równej 500(pokazaną na wykresie) jak i innych
- najbardziej optymalne wyniki występują dla kadencji równej $3n$, zgodnie z literaturą podaną na wykładzie [3] czyli 150 zarówno dla liczby iteracji równej 500(pokazaną na wykresie) jak i innych
- licznik krytyczny ma duże znaczenie na wynik i dla liczby równej 20 jest on najbardziej optymalny
- zwiększenie liczby iteracji ma wpływ na poprawę wyniku dla wszystkich innych parametrów. Warto jednak zauważyć, że wraz z liniowym wzrostem liczby iteracji – poprawa następuje coraz wolniej

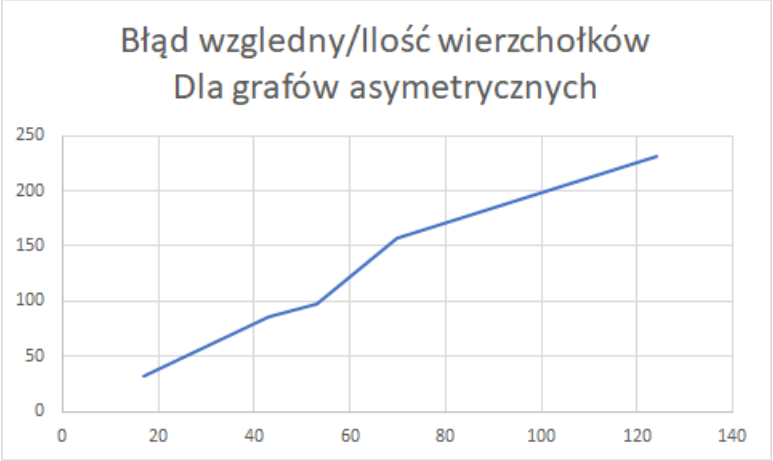
3.2 Zastosowanie znalezionych optymalnych parametrów dla grafów ze znanymi rozwiązaniami

Testy przeprowadzę dla dosyć dużych ilości iteracji, zależnych od problemu, licznikiem zdarzenia krytycznego równemu $\frac{i}{25}$, kadencji równej $3n$ oraz aspiracji równej $\frac{n}{5}$(lekko lepsze wyniki). Wszystkie testy wykonałem dla 1000 populacji dla każdego z grafów.

Nazwa pliku	Symetryczność	Błąd względny	Średni czas TS	Średni czas dynamicznego
gr17	symetryczny	13.57%	0.0025	2.83433
gr21	symetryczny	24.12%	0.0049	86.3032
gr24	symetryczny	34.59%	0.0077	1018.76
bays29	symetryczny	39.56%	0.0137	Brak danych
hk48	symetryczny	70.2%	0.6421	Brak danych
gr120	symetryczny	180.4%	11.70	Brak danych
br17	asymetryczny	32.35%	0.0025	2.8666
p43	asymetryczny	85.9%	0.537	Brak danych
ft53	asymetryczny	98.24%	0.925	Brak danych
ft70	asymetryczny	157.97%	2.31	Brak danych
kro124b	asymetryczny	270.9%	14.521	Brak danych



Rysunek 9: Wykres błędu względnego(oś X) od ilości wierzchołków(oś Y) na podstawie powyższej tabeli dla grafów symetrycznych



Rysunek 10: Wykres błędu względnego(oś X) od ilości wierzchołków(oś Y) na podstawie powyższej tabeli dla grafów asymetrycznych

4 Wnioski

Błąd względny wyników zbadanych instancji są mniej więcej liniowe zatem zakładam że implementacja jest poprawna. Do rozwiązania optymalnego poszczególnych instancji algorytm powinien zostać specjalnie dostosowany do tej instancji.

Z tabeli wyników widać że wyniki zgadzają się z teoretyczną złożonością (są mniej więcej 5-6 razy większe). Nieścisłości mogą wynikać, losowości wyników, niedokładności pomiaru czasu lub z tego, że niektóre instancje mają więcej minimów lokalnych od innych.

Z testów widać, że działanie dla grafu symetrycznego i asymetrycznego nie zmienia znacznie czasu działania algorytmu.

Średni czas już dla małych instancji jest aż 1000-krotnie mniejszy niż w przypadku programowania dynamicznego. W przypadku większych instancji różnica ta jest bardziej widoczna.

5 Bibliografia

1. Charles E. Leiserson. Ronald L. Rivest. Clifford Stein. *Introduction to Algorithms.* Third Edition. The MIT Press. Cambridge, Massachusetts London, England

2. A. Janiak, *Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów*, PLJ 1999.

3. J. Knox, *Tabu search performance on the symmetric traveling salesman problem*, 1994