Politechnika Wrocławska
14.01.2020r

Projektowanie Efektywnych Algorytmów
Etap 3

Temat:
Rozwiązanie TSP za pomocą algorytmów genetycznych

Prowadzący
dr inż. Tomasz Kapłon

Autor
Paweł Parczyk

Wstęp

Początki problemu Komiwojażera (TSP - Travelling salesman problem) sięgają XIX wieku, kiedy to William Rowan Hamilton opisał problem istnienia cyklu n-elementowego w grafie o n wierzchołkach. Za pierwszego autora, który sformalizował matematycznie problem komiwojażera, uznaje się Karla Mengera, który w 1930 zwrócił szczególną uwagę na trudność w obliczeniu optymalnego rozwiązania. Z uwagi na prostotę opisu problemu i trudność optymalizacji algorytmów służących do jego rozwiązania, problem stał się bardzo popularny.

Celem projektu było stworzenie algorytmów genetycznych rozwiązujących problem TSP dla instancji, których metody dokładne, zaimplementowane na poprzednim etapie projektu nie były w stanie odnaleźć.

Program został napisany w języku c++ w wersji 11 z wykorzystaniem technik programowania obiektowego. Kod został skompilowany przez kompilator *g++*, a następnie uruchomiony na komputerze zarządzanym przez system *Linux Ubuntu 16.04*, sama maszyna wyposażona była w 8GB pamięci ram i procesor *intel core i5*.

Algorytmy zostały napisane w wersjach iteracyjnych w celu zminimalizowania czasu wykonania.

Cechy metod niedokładnych

Podczas realizacji pierwszego etapu zostały zaimplementowane metody, których zadaniem było znalezienie najlepszego rozwiązania (dla rozpatrywanego problemu najlepszy to taki dla którego koszt przejścia jest najniższy). Z uwagi cechy problemu TSP (liczba potencjalnych rozwiązań wynos *n!*) takie algorytmy mają ograniczenia.

Problem z uzyskaniem wyniku optymalnego pojawia się gdy czas przetwarzania staje się dłuższy niż maksymalny dopuszczony czas, lub maszyna przetwarzająca problem ma mniej pamięci niż algorytm potrzebuje.

Problem skończonych zasobów czasowych i sprzętowych rozwiązują algorytmy niedokładne. Ich zadaniem jest przeszukiwanie przestrzeni stanów przez określony czas i znalezienie jak najlepszego rezultatu w tym czasie. Daje to możliwość przetwarzania znacznie większych instancji problemów, kosztem dokładności rozwiązań. Pomimo zmiany strategii znalezienia optymalnego rozwiązania w dalszym ciągu pozostaje możliwe.

Algorytmy genetyczne

Algorytmy genetyczne są rodzajem heurystyki przeszukującej przestrzeń rozwiązań w celu znalezienia najlepszego z nich.

Algorytmy genetyczne swoją nazwę zawdzięczają zjawisku ewolucji biologicznej z której jej twórcy czerpali inspirację. W procesie projektowania definiuje się środowisko, w którym istnieje populacja. Każdy osobnik tej populacji ma przypisany pewien zbiór informacji, jest to jego genotyp, który jest podstawą do stworzenia fenotypu (genotyp jest zestawem cech dziedziczonych, a fenotyp to zespół cech organizmu wynikający z genotypu). Osobniki na podstawie fenotypu poddaje się ocenie przez wyznaczenie wartości funkcji przystosowania modelującej środowisko – genotyp określa rozwiązanie, a funkcja przystosowania ocenia jak dobre ono jest.

Genotyp składa się z chromosomów, a każdy chromosom składa się z genów.

Działanie zaimplementowanego algorytmu można opisać poniższą listą kroków:

1. Losuje się początkowa **populację**.
2. Populacja zostaje poddana ocenie przystosowania, w tym procesie jest wykorzystywana **funkcja dopasowania**, najlepsze osobniki przechodzą dalej do procesu reprodukcji.
3. Wyselekcjonowane osobniki zostają poddane reprodukcji, w wyniku której tworzone jest kolejne pokolenie.
    1. Osobniki zostają poddane **krzyżowaniu**.
    2. Na uzyskanych osobnikach zostaje wykonana **mutacja**.
4. Jeżeli nie nastąpił warunek zakończenia to powrót do 3.

Algorytm wymaga określenia kilku parametrów, które wypływają na jakość otrzymywanych rozwiązań. Pierwszym z nich jest **czas przetwarzania**. Wpływa on na długość przetwarzania a w konsekwencji na liczbę pokoleń, która zostanie wygenerowana. Kolejnym jest **wielkość populacji**, która określa ilość osobników w każdym pokoleniu. Trzecim parametrem jest **operator krzyżowania**. Jest to funkcja określająca sposób łączenia osobników rodzicielskich w celu uzyskania osobników potomnych. Następnym parametrem jest **funkcja mutacji**, która służy do wprowadzenia do osobników pewnych cech losowych w celu zwieszenia różnorodności osobników. Ostatnim parametrem jest **prawdopodobieństwo wystąpienia mutacji**.

Oprócz operacji na samych osobnikach, należy jeszcze zdefiniować środowisko, w którym żyją generowane osobniki. Robi się to projektując **funkcję dopasowania**, która ocenia jak dobrze dany osobnik jest przystosowany do życia w symulowanym środowisku. Dla rozpatrywanego problemu funkcja dopasowania oblicza długość ścieżki jaką reprezentuje dany osobnik. Im mniejsza tym osobnik jest lepiej przystosowany.

Operatory krzyżowania

Edge Crossover (krzyżowanie krawędziowe): Technika krzyżowania chromosomów, w której wykorzystuje się już istniejące połączenia pomiędzy chromosomami (*krawędzie*).
Prezentacja operatora dla przykładowej pary rodziców:

- Rodzic 1: 1 2 6 5 4 7 3
- Rodzic 2: 7 6 1 2 3 4 5

Na początku wyznaczone zostają pary sąsiadujących ze sobą genów u obu rodziców i zapisuje się je bez powtórzeń w liście sąsiadów. Sąsiadują ze sobą takie geny które mają wspólną krawędz.

- 1: 3 2 6 (u rodzica 1 liczba *1* sąsiaduje z *7* i *2*, a u rodzica 2 liczba *1* sąsiaduje z *6* i *2*)
- 2: 1 6 3
- 3: 1 7 2 4
- 4: 5 7 3
- 5: 4 6 7
- 6: 7 1 2 5
- 7: 4 3 5 6

Następnie losowo wybiera się jeden z genów i ustawia się go na pierwszej pozycji u potomka.

- Potomek: 6 [x] [x] [x] [x] [x] [x]

Wybrany gen usuwa się z listy sąsiadów, a następnie usuwa się sąsiedztwa pozostałych genów z nim. Następnym krokiem jest znalezienie takiego genu, dla którego lista sąsiadów jest najmniejsza. Jeżeli występuje więcej genów o równej długości to wybrany zostaje losowy z nich. Operację tą powtarza się aż do uzupełnienia wszystkich genów u potomka.

Partially Mapped Crossover (krzyżowanie z częściowym odwzorowaniem): Technika krzyżowania polegająca na wybraniu pewnego ciągu genów u obu rodziców, a następnie zamianę tych części u pary rodziców. Następnie pozostałe geny zamienia się w taki sposób aby osobnik spełniał wymagania środowiska.
Prezentacja operatora dla przykładowej pary rodziców:

- Rodzic 1: 1 2 6 5 4 7 3
- Rodzic 2: 1 6 7 2 3 4 5

W wyniku działania operatora powstają dwa osobniki potomne *R* i *S*. Losowo zostaje wybrany fragment osobników, który zostanie zamieniony.

- R: [x] [x] 7 2 3 4 [x] – Wprowadzony zostaje ciąg od rodzica 2
- S: [x] [x] 6 5 4 7 [x] – Wprowadzony zostaje ciąg od rodzica 1

Następnie osobniki zostają uzupełnione genami, które nie powodują konfliktu

- R: 1 [x] 7 2 3 4 [x] – Wprowadzone zostają geny od rodzica 1
- S: [x] [x] 6 5 4 7 [x] – W tym przypadku każdy z genów (*7*, *6*, *5*) powoduje konflikt

W następnym kroku wstawiane są geny zgodnie z odwzorowaniem: 7 ⇔ 6, 2 ⇔ 5, 3 ⇔ 4, 4 ⇔ 7.

- R: 1 5 7 2 3 4 [x]
- S: 1 [x] 6 5 4 7 2

Jeśli nie udało się wstawić niektórych genów potrzebne jest podwójne lub kolejne odwzorowanie odwzorowanie:

- R: 1 5 7 2 3 4 6
- S: 1 3 6 5 4 7 2

Aby wstawić *6* w *R* wykorzystano odwzorowanie 3 ⇔ 4 ⇔ 7 ⇔ 6, natomiast do wstawienia *3* w *S* został użyty ciąg odwzorowań 6 ⇔ 7 ⇔ 4 ⇔ 3. W konsekwencji zaprezentowanego cyklu przetwarzania powstały dwa osobniki potomne R(1,5,7,2,3,4,6) i S(1,3,6,5,4,7,2). Aby algorytm genetyczny był kompletny niezbędne jest wprowadzenie mechanizmu mutacji.

Algorytmy mutacji

Mutacje polegają na wprowadzeniu losowych zmian w obrębie jednego genu lub kilku z nich, bez znacznej zmiany osobnika. Celem użycia mutacji jest zapewnienie zmienności osobników, stwarzają one możliwość wyjścia z optimów lokalnych. Cechą odróżniającą mutacje od krzyżowania jest to, że działają na pojedynczym osobniku i zachodzą z pewnym prawdopodobieństwem.

Zamiana dwóch genów miejscami: Metoda ta polega na wybraniu dwóch losowych genów danego osobnika i zamianie ich pozycjami. Dla zaprezentowanego wyżej osobnika:

- R: 1 5 7 2 3 4 6

mutacja genów o indeksach 3 i 5 polegałaby na zamianie liczb *2* z *4*, co poskutkowałoby otrzymanie następującego osobnika:

- R: 1 5 7 4 3 2 6

W przypadku rozwiązywania problemu TSP tak zmodyfikowana ścieżka która jest reprezentowana przez osobnika różni się jedynie dwiema krawędziami.

Zamiana dwóch krawędzi miejscami: Metoda ta polega na wybraniu dwóch krawędzi pomiędzy genami i zamianie ich miejscami. Krawędź definiuje się podobnie ja w opisie *krzyżowania krawędziowego*. Dla zaprezentowanego wyżej osobnika:

- R: 1 5 7 2 3 4 6

wybrane zostają dwie krawędzie (pomiędzy genami o indeksach 2, 3 i pomiędzy 4,5). Po przeprowadzeniu mutacji osobnik R zostanie przekształcony do postaci

- R: 1 3 4 2 5 7 6

Algorytm mutacji krawędzi wprowadza większe zmiany w strukturze osobnika niż algorytm zamiany genów ponieważ przestawiane są 4 geny a zatem zmieniają się dokładnie 4 krawędzie.

Optymalizacja parametrów

Zaprojektowany algorytm genetyczny wraz z operatorami krzyżowania i mutacji wymagał odnalezienia parametrów optymalnych pracy, dla których czas stosunek kosztów (czas przetwarzania) do rezultatów (jakości odnajdowanych rozwiązań) będzie zadowalający.

Proces testowania

Każda z parametrów optymalizowano dla każdej dostępniej kombinacji operatorów krzyżowania i mutacji. Do rozróżnienia testowanych układów użyto oznaczeń

- **EC –** Edge Crossover – operator krzyżowania krawędziowego,
- **PMX** - Partially Mapped Crossover **–** operator krzyżowania z częściowym odwzorowaniem,
- **EM –** Edge Mutation - operator mutacji krawędziowej,
- **GM** – Gene Mutation – operator mutacji genu.

Poza optymalizacją parametrów celem testowania było również określenie najlepszej kombinacji algorytmów.

Do testów użyto instancji problemu dostępnych na stronie http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

- Graf 42 wierzchołkowy *Swiss42*, z optymalną ścieżką długości 1273
- Graf 17 wierzchołkowy Bays29, z optymalną ścieżką długości 2020

Optymalizacja czasu przetwarzania

Pierwszym parametrem poddanym optymalizacji był czas przetwarzania. Przeprowadzono 20 testów dla ustalonych parametrów a następnie uśredniono wyniki. Na *rysunku 1* zaprezentowano przykładową tabele wyników przetwarzania PMX i EM grafu o 42 wierzchołkach. Tabela zawiera informacje o znalezionej drodze i błędzie względnym.

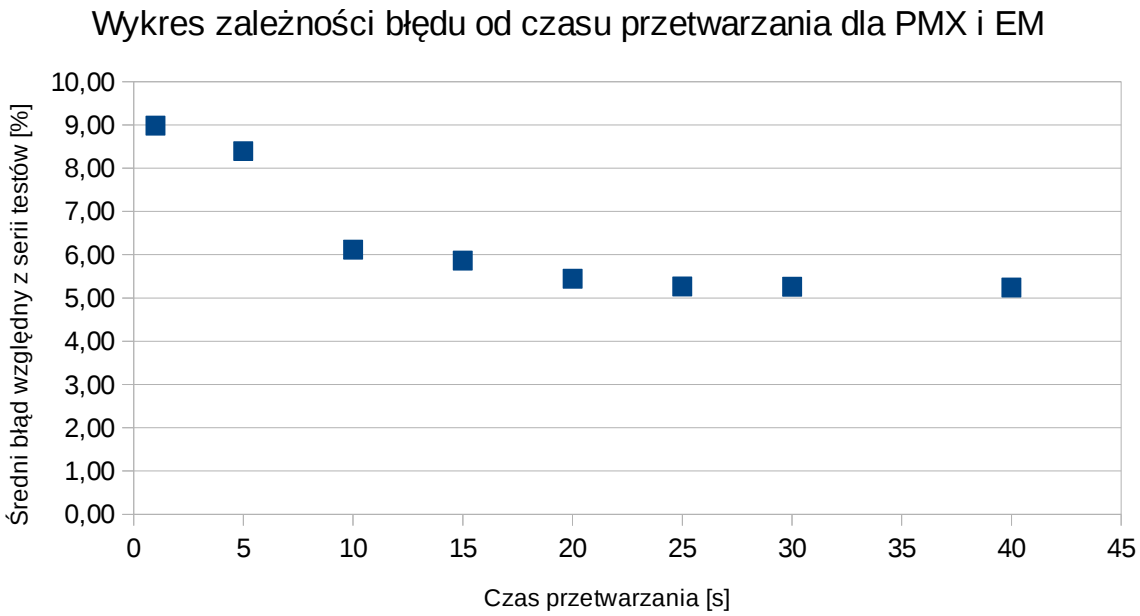
Przetwarzanie grafu				
liczba wierzchołków				42
czas przetwarzania				1000 [ms]
operator krzyżowania				PMX
operator mutacji				EM
rozwiązanie optymalne				1273
l.p.	**czas [ms]**	**droga**	**czas [s]**	**Błąd [%]**
1	1000,00	1364	1,00	6,67
2	1000,00	1440	1,00	11,60
3	1000,00	1384	1,00	8,02
4	1000,00	1410	1,00	9,72
5	1000,00	1404	1,00	9,33
6	1000,00	1394	1,00	8,68
7	1000,00	1388	1,00	8,29
8	1000,00	1383	1,00	7,95
9	1000,00	1407	1,00	9,52
10	1000,00	1407	1,00	9,52
11	1000,00	1386	1,00	8,15
12	1000,00	1404	1,00	9,33
13	1000,00	1383	1,00	7,95
14	1000,00	1384	1,00	8,02
15	1000,00	1402	1,00	9,20
16	1000,00	1383	1,00	7,95
17	1000,00	1375	1,00	7,42
18	1000,00	1396	1,00	8,81
19	1000,00	1465	1,00	13,11
20	1000,00	1420	1,00	10,35

Rysunek 1: wyniki jakości przetwarzani grafu (42 wierzchołkowego) algorytmem genetycznym PMX i EM.

Dokładnie w ten sam sposób przetestowano następujące czasy przetwarzania: 1, 5, 10, 15, 20, 25, 30, 40 sekund. *Rysunek 2* przedstawia zależność średniego błędu względnego, który jest wyznacznikiem jakości algorytmu od czasu przetwarzania.

Czas przetwarzania	**1**	**5**	**10**	**15**
Średni błąd [%]	8,98	8,39	6,11	5,87
Czas przetwarzania	**20**	**25**	**30**	**40**
Średni błąd [%]	5,45	5,26	5,26	5,24

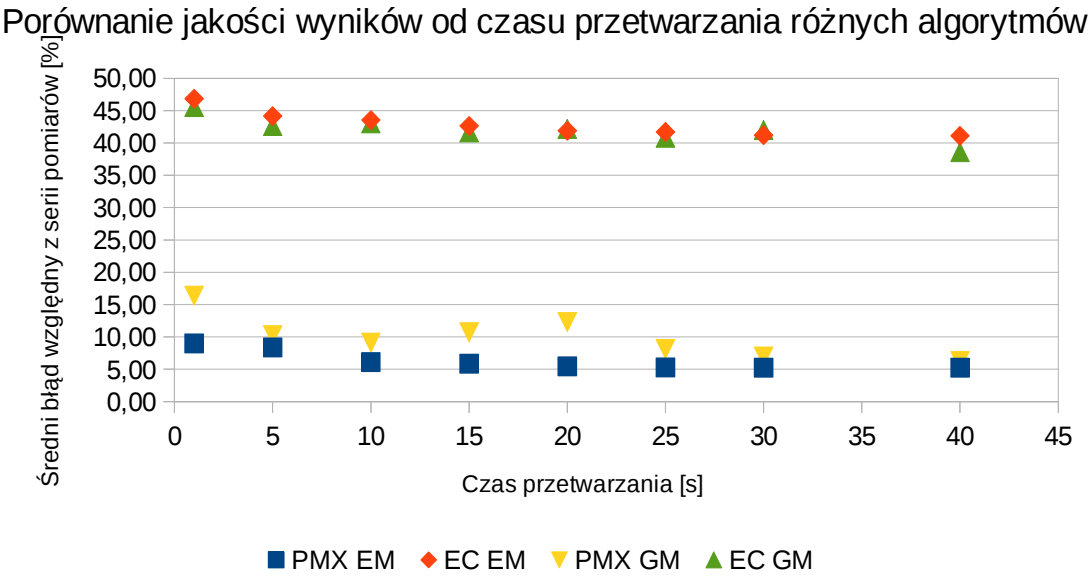
Rysunek 2: Tabela zależności jakości wyników od czasu przetwarzania



Rysunek 3: Wykres zależności jakości wyników od czasu przetwarzani.

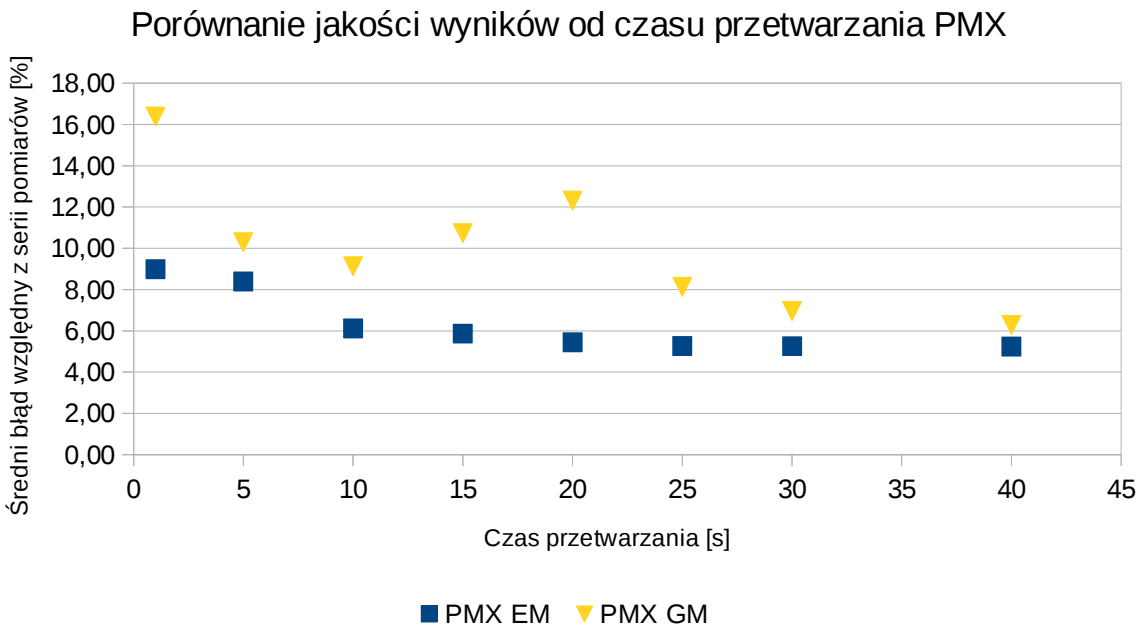
Wykres zaprezentowany na *rysunku 3*, przedstawia zależność jakości wyników od czasu przetwarzania. Uzyskane rezultaty pokrywają się z oczekiwaniami. Dłuższy czas przetwarzania powoduje że algorytm może wygenerować większą ilość pokoleń, przez co sprawdzi więcej możliwych rozwiązań problemu, co zwiększa prawdopodobieństwo znalezienia lepszych wyników. Jednakże zależność ta nie jest liniowa.

W dalszej części analizie poddane zostały pozostałe kombinacje operatorów krzyżowania i mutacji, a rezultaty przedstawiono na *rysunku 4*.

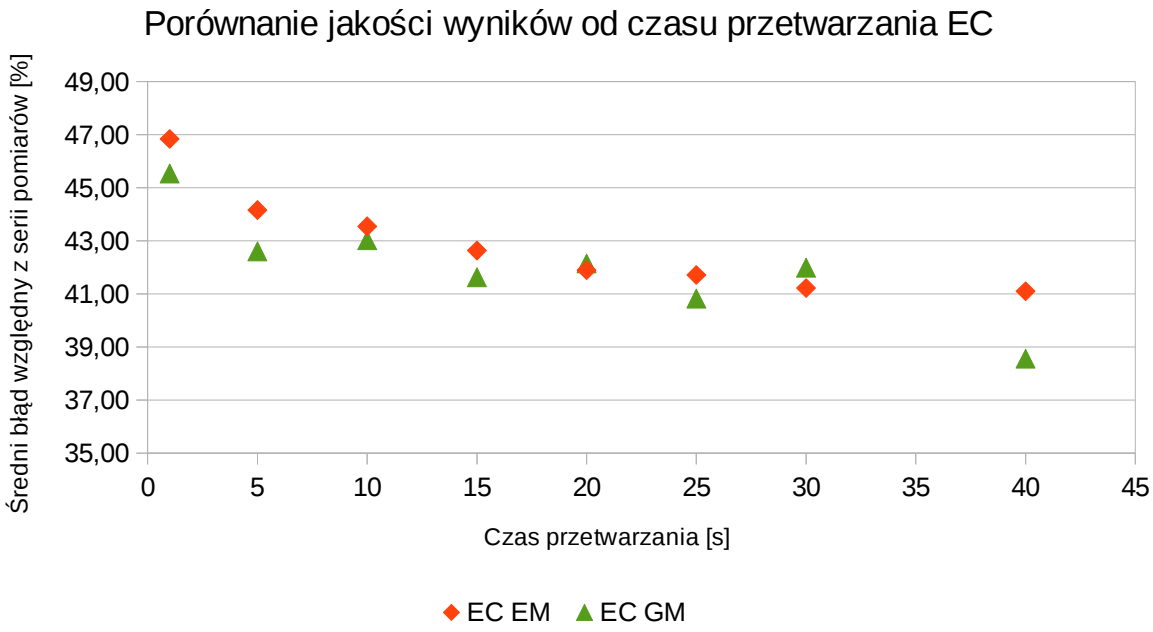


Rysunek 4: Wykres zależności jakości wyników od czasu przetwarzania dla różnych algorytmów.

Z wykresu na *rysunku 4*, można wyciągnąć wniosek że algorytm PMX jest skuteczniejszy od algorytmu EC. Niezależnie od czasu jaki był przeznaczony na przetwarzanie i algorytmu mutacyjnego jego wyniki są 4-krotnie lepsze.



Rysunek 5: Wykres zależności jakości wyników od czasu przetwarzania dla PMX



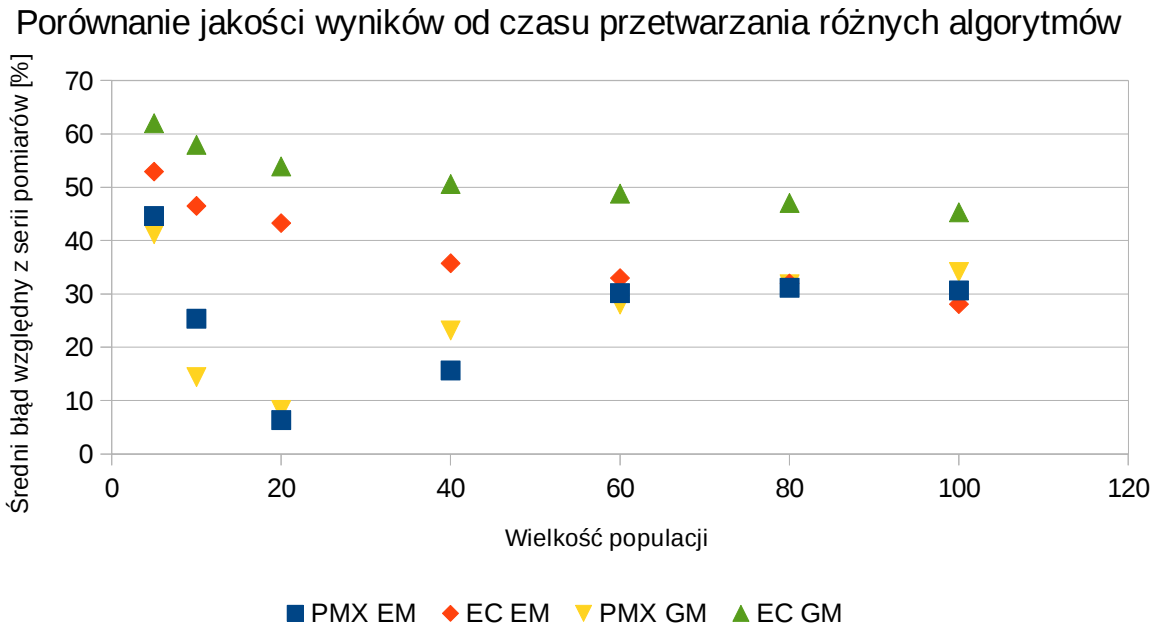
Rysunek 6: Wykres zależności jakości wyników od czasu przetwarzania dla EC

Rysunek 5 i *rysunek 6* prezentują porównanie pomiędzy różnymi metodami mutacji w przy zastosowaniu tego samego mechanizmu krzyżowania.

Na podstawie przeprowadzonych pomiarów określono, że najbardziej odpowiednim czasem przetwarzania jest **15 sekund**. Jest to czas podczas którego krzywe zbliżają się swoim kształtem do funkcji stałych – zysk z wydłużania czasu będzie już nieznaczny. Natomiast czas przetwarzania przy wykonywaniu pomiarów zgodnie z założeniami nie jest jeszcze długi.

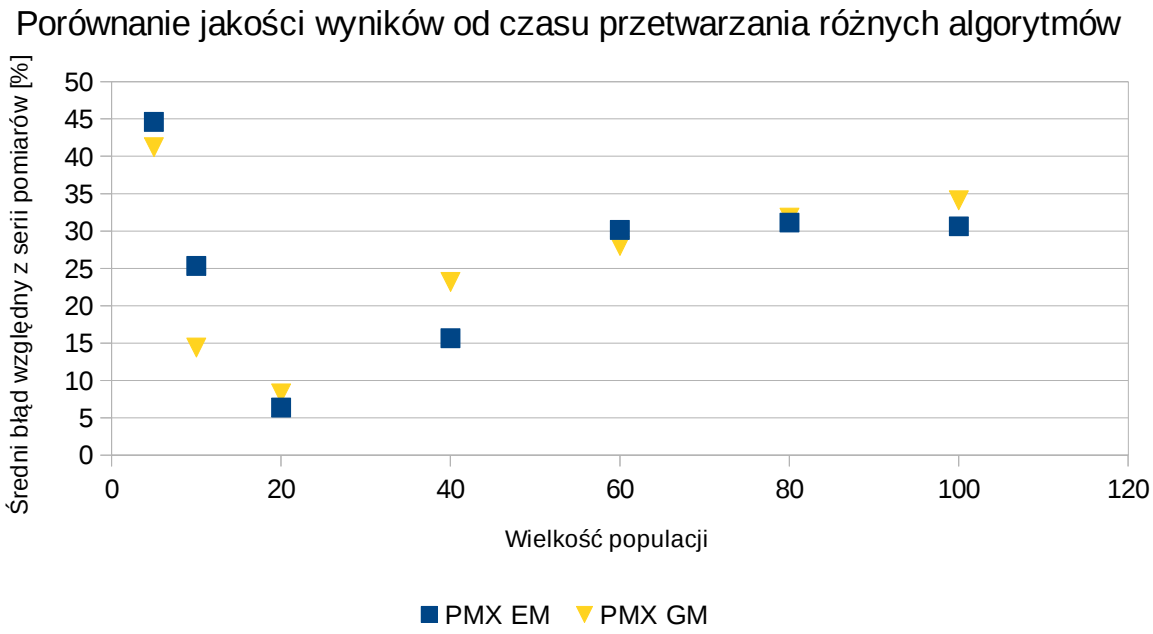
Optymalizacja wielkości populacji

W dalszych testach zastosowano wybrany czas do przetwarzania (15 sekund) i wybierano kolejno populacje o wielkościach 5, 10, 20, 40, 60, 80, 100 osobników. Uzyskane rezultaty zaprezentowano na *rysunku 7*.



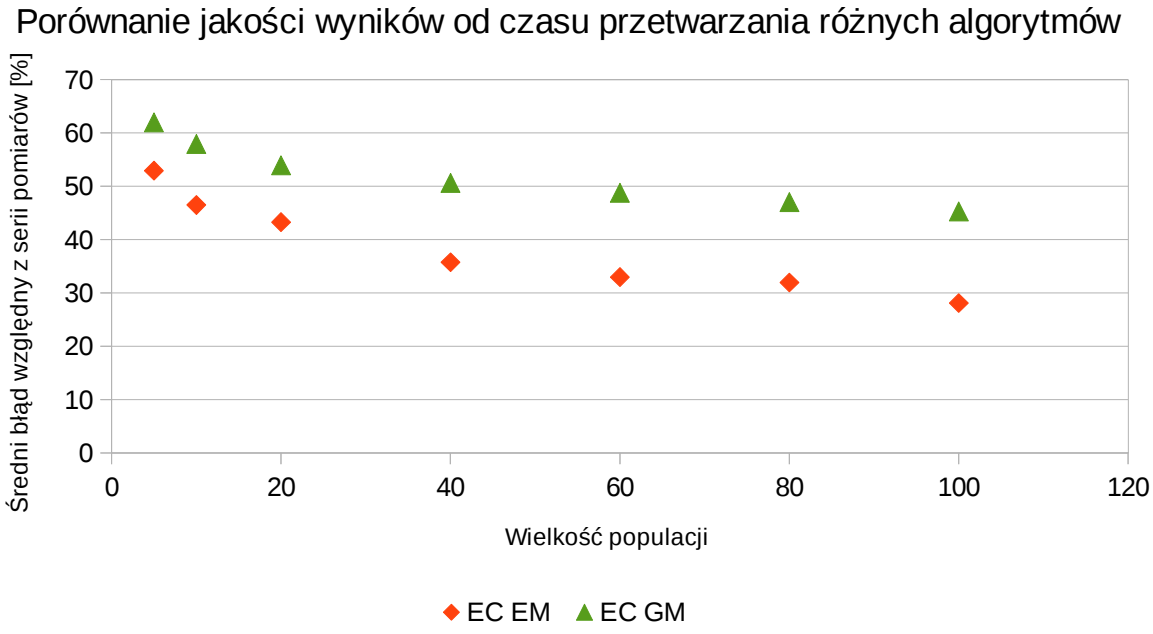
Rysunek 7: Porównanie jakości wyników różnych algorytmów w zależności od wielkości populacji

Z wykresu na *rysunku 7*, widać zależność iż PMX jest skuteczniejszym algorytmem od EC aczkolwiek, przy większych wielkościach populacji tendencja ta zaczyna się odwracać. Algorytm EC EM wraz ze wzrostem populacji cały czas poprawia wyniki, natomiast algorytmy PMX tracą na dokładności. Dla 100 EC jest już lepszy od PMX.



Rysunek 8: Porównanie jakości wyników dla algorytmu PMX przy różnych wielkościach populacji

Analiza zachowania się jakości wyników (*Rysunek 8*) dla różnych wielkości populacji dla algorytmu PMX wskazuje, że algorytm najlepiej zachowuje się gdy wielkość populacji wynosi 20 osobników. Warto zauważyć również, że nie ma na to wpływu mechanizm mutacyjny. Aczkolwiek widać że zasada zauważona przy zmianach czasu operacji przestaje działać. Metoda mutacji genowej (GM) nie jest już zawsze gorsza od metody EM.



Rysunek 9: Porównanie jakości wyników dla algorytmu EC przy różnych wielkościach populacji

Wykres na *rysunku 9* przedstawia zależność jakości wyników od wielkości populacji dla różnych metod mutacji. Metoda mutacji genów daje znacznie gorsze wyniki od metody mutacji krawedzi, a różnica ta rośnie wraz ze wzrostem populacji. Jest to odwrócenie tendencji z pomiarów dla różnych czasów przetwarzania. Zachodzi podobne zjawisko jak dla algorytmów PMX, choć jest ono znacznie bardziej widoczne.

Z uwagi na bardzo odległe charakterystyki zmian od różnych wielkości populacji do dalszego analizowania zostały wybrane wielkości **10, 20, 100** osobników

Optymalizacja prawdopodobieństwa zajścia mutacji

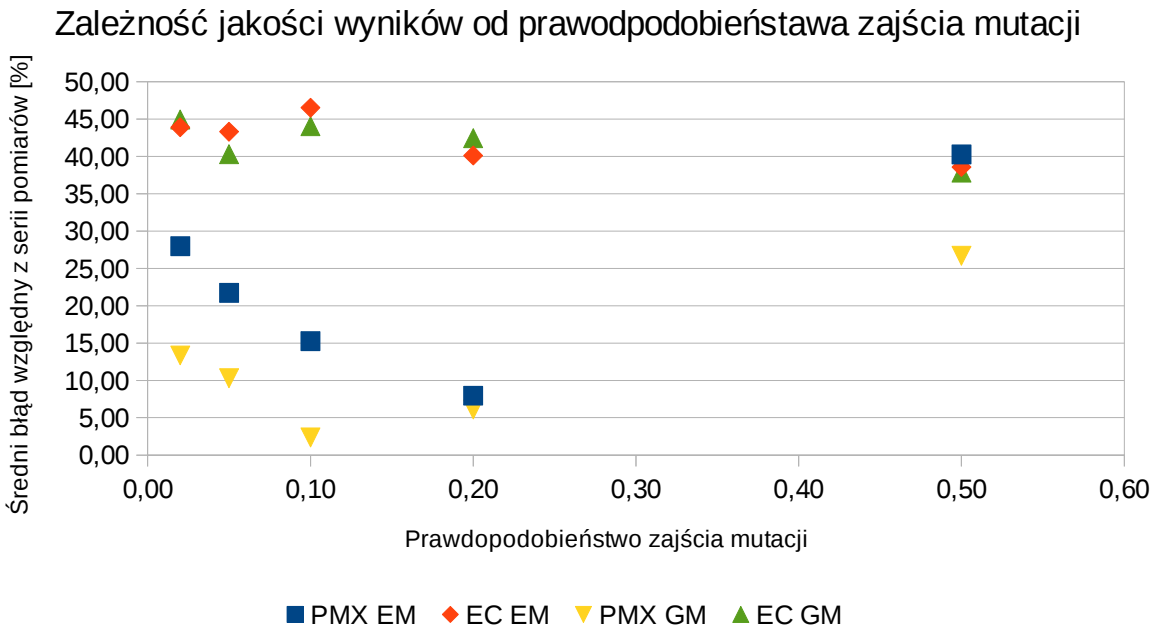
Podobnie jak w zjawisku ewolucji biologicznej prawdopodobieństwo zajścia mutacji jest niskie. Mutacja w algorytmie genetycznym ma za zadanie zwiększyć szansę na wyjścia z minimów lokalnych funkcji optymalizowanej. W zależności od wartości prawdopodobieństwa mutacje mogą polepszyć wyniki jak i je pogorszyć. Dlatego niezbędne jest poprawne dobranie parametrów.

Do prób wybrano prawdopodobieństwa **0,02; 0,05; 0,1; 0,2; 0,5;** i zastosowana znalezione wcześniej optymalne wartości wielkości populacji i czasu przetwarzania które wynosiły odpowiednio 20 osobników i 15 sekund.

	0,02	**0,05**	**0,10**	**0,20**	**0,50**
PMX EM	27,96	21,71	15,25	7,95	40,26
EC EM	43,90	43,30	46,51	40,09	38,56
PMX GM	13,34	10,29	2,38	6,12	26,67
EC GM	44,94	40,29	44,04	42,42	37,81

Rysunek 10: Zależność jakości wyników przetwarzania od prawdopodobieństwa zajścia mutacji

Na podstawie tabeli zawartej na *rysunku 10* wygenerowano wykres który został zaprezentowany na rysunku 11.



Rysunek 11: Zależność jakości wyników przetwarzania od prawdopodobieństwa zajścia mutacji

Na rysunku 11 zaprezentowano wykres zależności jakości otrzymywanych wyników od prawdopodobieństwa zajścia mutacji. Algorytmy PMX i EC w inny sposób reagują na zachodzące mutacje. PMX osiąga najlepsze wyniki dla prawdopodobieństwa mieszczącego się pomiędzy 0,1, a 0,2 natomiast w przebiegu zależności dla algorytmy EC nie ma znaczących spadków wartości co może sugerować, że nie same mutacje nie wpływają ani pozytywnie ani negatywnie na przetwarzanie, muszą one zachodzić aby spełnić swoją rolę.

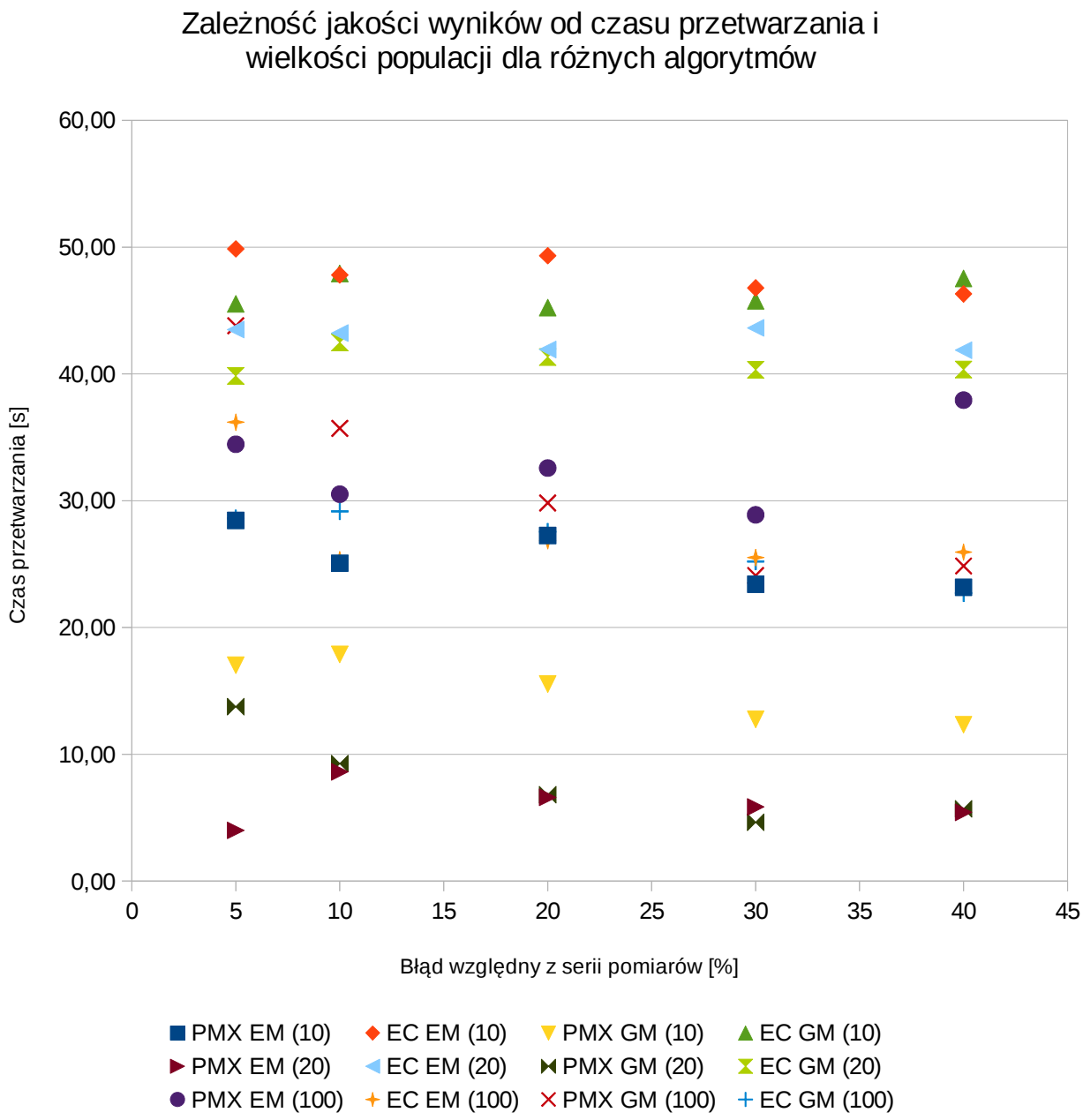
Analiza jakości wyników dla różnych populacji przy zmiennym czasie przetwarzania

Z wyników przetwarzania przy stałym czasie wybrano 3 wielkości populacji i dla każdej z nich wykonano testy zmiennej długości przetwarzania. Rezultaty zostały przedstawione na rysunku

	5	10	20	30	40
PMX EM (10)	28,44	25,07	27,26	23,41	23,17
EC EM (10)	49,86	47,81	49,32	46,78	46,31
PMX GM (10)	17,02	17,87	15,53	12,75	12,33
EC GM (10)	45,53	47,91	45,22	45,76	47,53
PMX EM (20)	4,00	8,61	6,60	5,84	5,42
EC EM (20)	43,50	43,22	41,93	43,62	41,87
PMX GM (20)	13,75	9,27	6,81	4,64	5,70
EC GM (20)	39,84	42,48	41,34	40,32	40,35
PMX EM (100)	34,45	30,51	32,57	28,88	37,93
EC EM (100)	36,19	25,34	26,84	25,51	25,95
PMX GM (100)	43,80	35,71	29,82	24,09	24,85
EC GM (100)	28,60	29,16	27,55	25,21	22,71

Rysunek 12: Tabla pomiarowa prezentująca błąd względny od czasu przetwarzania i wielkości populacji dla różnych algorytmów

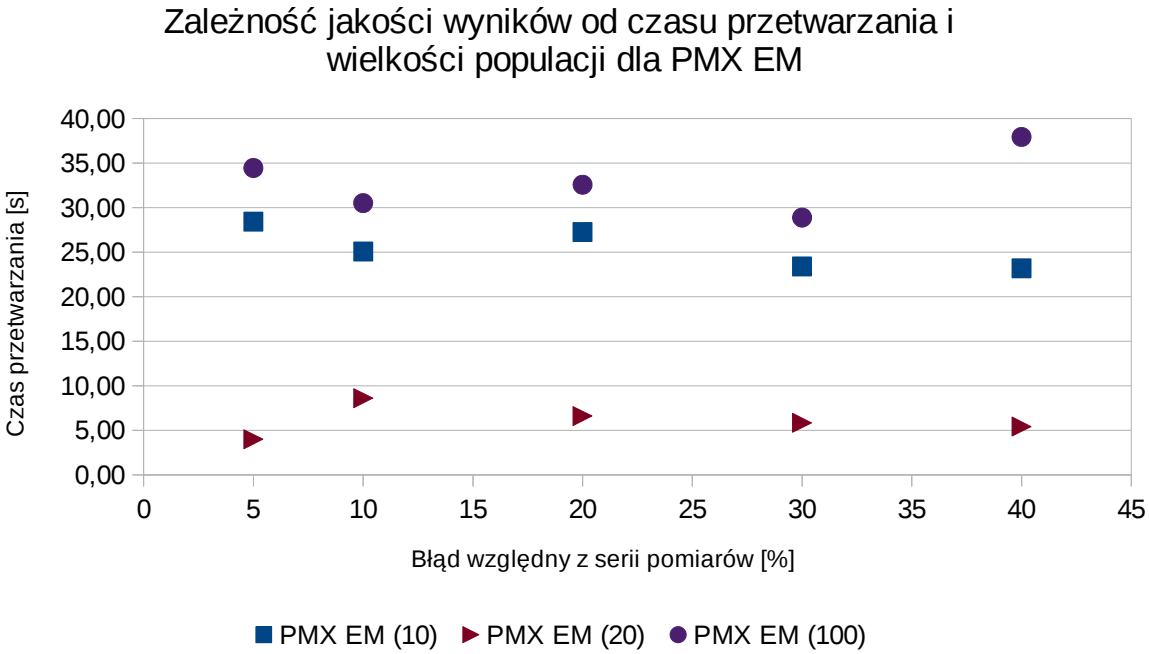
Na podstawie danych zawartych w tabeli na *rysunku 12* wygenerowano wykres tej samej zależności.



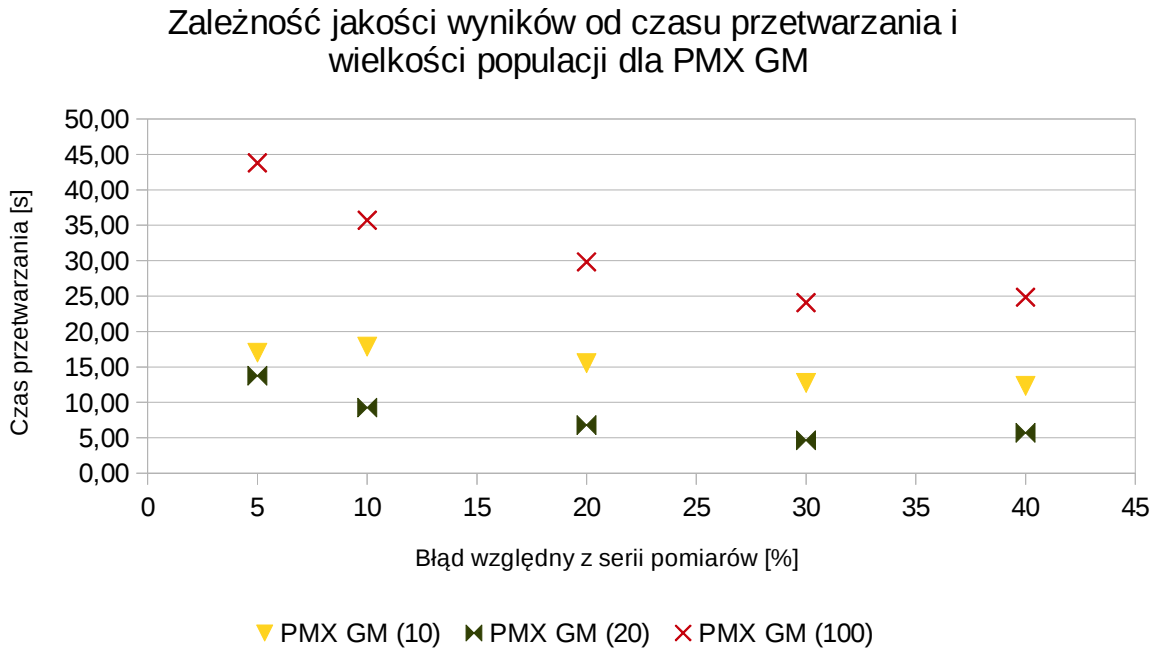
Rysunek 13: Wykres prezentujący błąd względny od czasu przetwarzania i wielkości populacji dla różnych algorytmów

Wykres zaprezentowany na *rysunku 13* przedstawia zależności, które pozwalają porównać różne parametry przetwarzania. Dzięki temu można określić parametry optymalne pracy poszczególnych algorytmów.

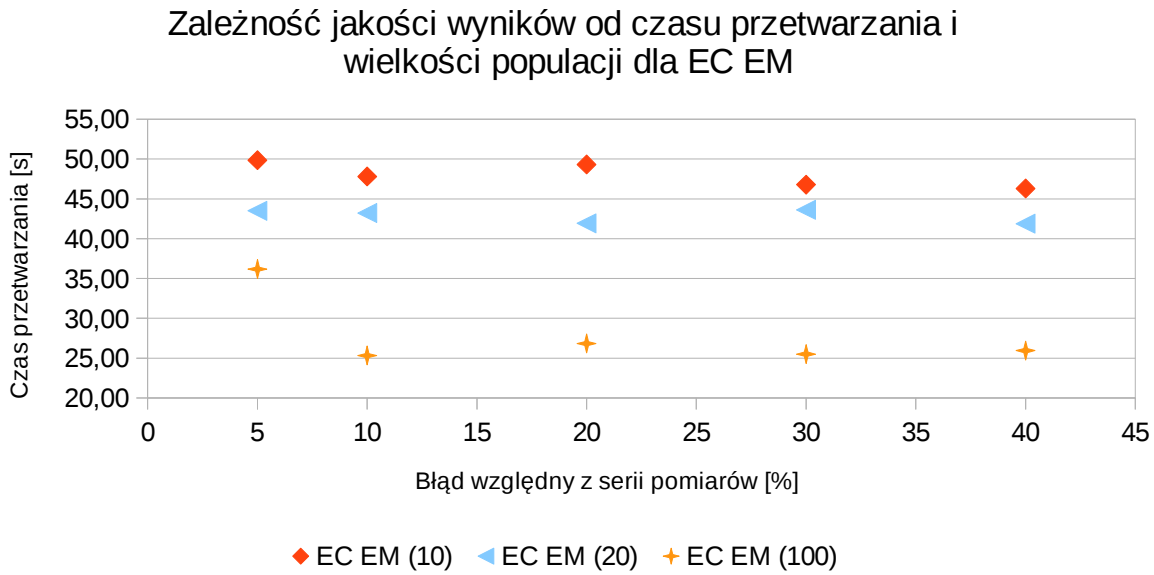
Najlepszym ze wszystkich algorytmów jest algorytm PMX z metodą mutacji EM przy wielkości populacji równej 20. Natomiast najgorszym jest EC z metodą mutacji EM przy 10 osobnikach w populacji.



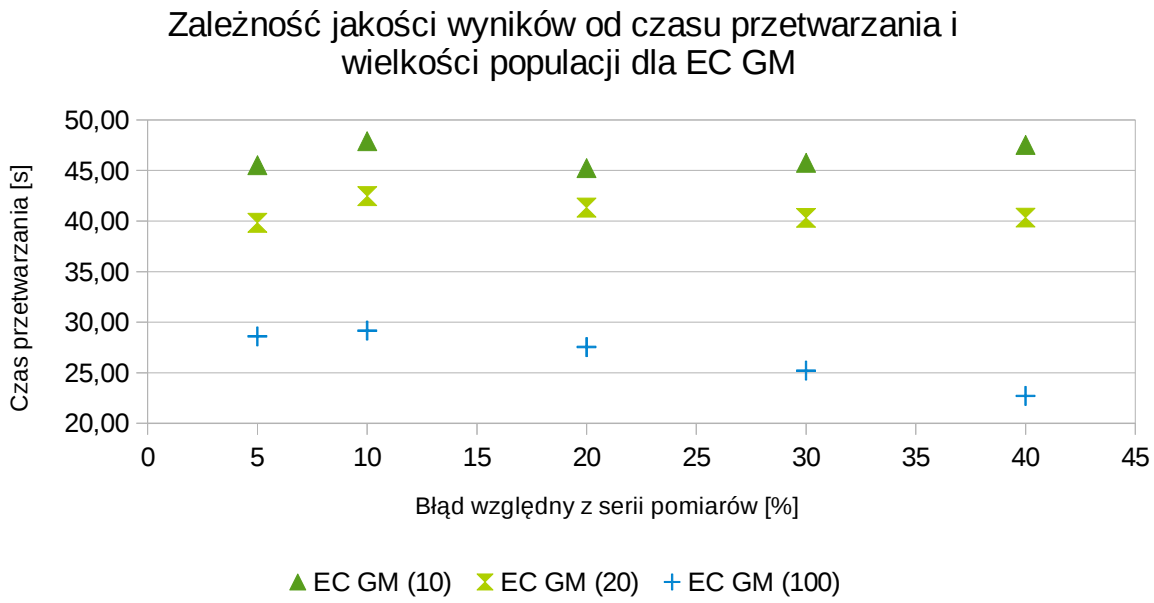
Rysunek 14: wykres prezentujący błąd względny od czasu przetwarzania i wielkości populacji dla algorytmu PMX EM



Rysunek 15: wykres prezentujący błąd względny od czasu przetwarzania i wielkości populacji dla algorytmów PMX GM



Rysunek 16: wykres prezentujący błąd względny od czasu przetwarzania i wielkości populacji dla algorytmów EC ME

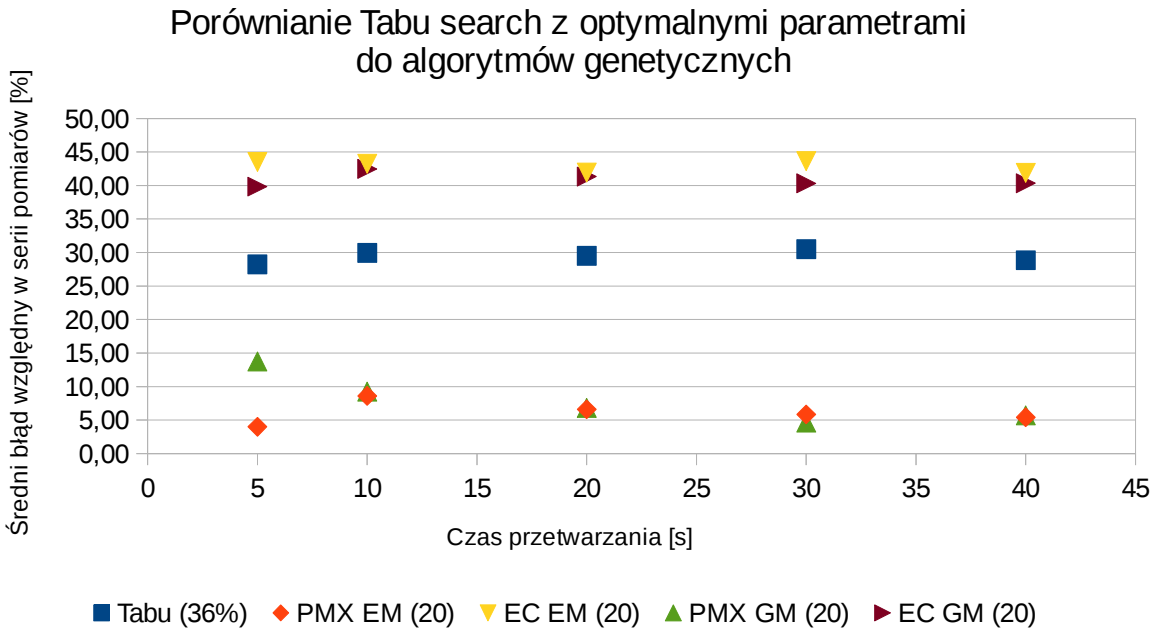


Rysunek 17: wykres prezentujący błąd względny od czasu przetwarzania i wielkości populacji dla algorytmów EC GM

Porównanie algorytmów genetycznych z algorytmem tabu search

Kolejnym etapem rozpatrywania algorytmów genetycznych było porównanie ich z wcześniej zaprojektowanymi i zrealizowanymi algorytmami. Podczas realizacji drugiego etapu został zaimplementowany algorytm realizujący metodę wyszukiwania z zakazami (**tabu search**). Dla zrealizowanie jego implementacji zostały określone parametry optymalne, gdzie wielkość listy wynosiła 38% wszystkich możliwych ruchów w instancji, a ilość błędów krytycznych została

określona na 20. Algorytmem rozwiązano instancję testową. Rezultat przetwarzania zaprezentowano na wykresie na *rysunku 18.*



Rysunek 18: Wykres porównanie zależności jakości wyników od czasu przetwarzania algorytmów genetycznych i tabu search

Rysunek 18 Przedstawia porównanie algorytmów genetycznych z tabu search. Wyniki przetwarzania wyszukiwania z zakazami plasuje się pomiędzy algorytmie PMX a algorytmem GM. Powyższe porównanie zostało przeprowadzone dla parametrów optymalnych określonych w poprzedzających badaniach i podczas realizacji drugiego etapu projektu.

Wnioski

Algorytmy generyczne są zbiorem algorytmów opartych w swoich założeniach o wiadomości zaczerpnięte z nauk przyrodniczych, wykorzystują mechanizmy, które w środowisku naturalnych odpowiadają za istnienie życia w formie jaką obecnie możemy oglądać. Jednakże pomimo prostego opisu samo wykorzystanie do rozwiązania problemu nie jest proste. Aby zaprojektować taki algorytm trzeba znać problem aby móc zdefiniować funkcję przystosowania i należy przeprowadzić badania aby określić parametry optymalne pracy – dopiero wtedy algorytmy genetyczne będą działać skutecznie.

Zaimplementowane w projekcie algorytmy rozwiązywały instancje problemu nieosiągalne dla algorytmów dokładnych w czasie relatywnie krótkim. Popełniały przy tym błędy na poziomie kilku procent, co jak najbardziej było do przyjęcia. Warto zauważyć, że niejednokrotnie znajdowane były rozwiązania optymalne.

Znalezione parametry optymalne:

- PMX: **20 osobników**, **15 sekund**, prawdopodobieństwo **0,15**
- EC: **100 osobników**, **15 sekund**, prawdopodobieństwo **0,2**