Politechnika Wrocławska

Projektowanie efektywnych algorytmów

Projekt nr 3:
Implementacja i analiza efektywności algorytmu genetycznego (ewolucyjnego) dla problemu komiwojażera

Autor:
Bartosz Pogoda 225988

Prowadzący:
dr inż. Jarosław Mierzwa

8 stycznia 2018

Spis treści

1 Wstęp teoretyczny

1.1 Opis problemu

Problem komiwojażera definiowany jest dla grafów pełnych. Wierzchołki grafu nazywane są miastami, a krawędziom przypisane są wartości reprezentujące odległości między nimi. Problem polega na odnalezieniu najkrótszej ścieżki, takiej aby odwiedzić każde miasto dokładnie raz i wrócić do punktu wyjścia.

Wiele problemów, występujących w sektorach takich jak transport oraz logistyka można sprowadzić do problemu komiwojażera. Zastosowanie efektywnych algorytmów rozwiązujących ten problem nie rzadko prowadzi do redukcji kosztów i optymalizacji pewnych rzeczywistych procesów. Przykładową aplikacją mogło by być wyznaczenie optymalnej trasy dla autobusu szkolnego, tak aby odwiedził wszystkie przystanki z dziećmi w jak najkrótszym okresie czasu. Wierzchołkami grafu byłyby poszczególne przystanki, a wartościami na krawędziach szacunkowy czas przemieszczenia się autobusu między przystankami. Warto tutaj zauważyć, że czas przemieszczania się z przystanku A do B, może różnić się od czasu przemieszczenia z B do A. Różnice te są uwzględnione w asymetrycznym problemie komiwojażera, który zostanie poddany analizie w tym sprawozdaniu.

1.2 Opis algorytmu

Algorytmy genetyczne (ewolucyjne) opierają się założeniach teorii ewolucji Darwina. Procesy dotyczące selekcji naturalnej występujące w rzeczywistym środowisku są symulowane w celu rozwiązania problemów optymalizacyjnych. Podstawową zasadą, która sugeruje skuteczność takich algorytmów jest fakt, że z biegiem czasu (w wyniku ewolucji oraz selekcji naturalnej) populacja staję się coraz lepiej przystosowana do środowiska, w którym żyje, o ile warunki środowiska diametralnie się nie zmieniają - co zazwyczaj nie ma miejsca w przypadku problemów, dla których tworzymy takie algorytmy.

1.3 Kroki algorytmu

Podstawowy algorytm genetyczny składa się z kroków opisanych poniżej. Etapy procesu (1.3.2 - 1.3.5) są powtarzane wielokrotnie aż do osiągnięcia kryterium stopu.

1.3.1 Inicjalizacja

Zostaje utworzona początkowa populacja, o określonej wielkości. Podczas inicjaizacji w obrębie projektu jedna ścieżka jest znajdowana za pomoca algorytmu zachłannego, a reszta generowana jest w sposób losowy.

1.3.2 Ewaluacja - wyznaczenie przystosowania

Wyznaczane są wartości przystosowania jednostek populacji. W tym celu definiuje się specjalną funkcję przystosowania (funkcja FIT), która mierzy jak "dobra" jest dana jednostka. Dla problemu komiwojażera wartość przystosowania jednostki-ścieżki określa się jako odwrotność jej długości.

W obrębie projektu osobnik najlepiej przystosowany jest bezpośrednio umieszczany w następnej generacji. Interpretacją biologiczną mogłoby tutaj być umieszczenie dobrze przystosowanego osobnika w inkubatorze na czas jednej generacji, w celu zapobiegnięcia modyfikacji jego materiału genetycznego.

1.3.3 Selekcja

W tym kroku następuje wybór z populacji jednostek, które zostaną rodzicami dla kolejnej generacji. Jedną z metod wyboru jest metoda turniejowa - organizowane są turnieje o określonej wielkości, do których dobiera się uczestników w sposób losowy. Turniej wygrywa najlepiej przystosowany osobnik (maksymalna wartość funkcji FIT). Warto tutaj zaznaczyć, że dobór wielkośći turnieju jest istotny. Małe turnieje zwiększają "dywersyfikację" przestrzeni rozwiązań, ponieważ zwiększają szansę jednostek "słabych", które jednak w wyniku krzyżowania mogą dać nowe, potencjalnie lepsze rozwiązania. Duże turnieje znacznie fawozyrują jednostki lepiej przystosowane, co może prowadzić do osiągnięcia minimum lokalnego.

1.3.4 Krzyżowanie

Osobniki wybrane w fazie selekcji są krzyżowane w celu wymieszania ich cech. W wyniku krzyżowania powstaje dziecko, które w idealnym przypadku odziedziczyłoby najlepsze cechy obojga rodziców. Definiowane jest prawdopodobieństwo z jakim odbędzie się krzyżowanie.

- Krzyżowanie PMX

  Operator PMX (Partially-Matched Crossover) gwarantuje brak powtórzeń genów (numerów miast) w powstałym chromosomie (w powstałej ścieżce). Podczas operacji krzyżowania w pierwszej kolejności losowany jest pewien zakres kolejnych pozycji ścieżki. Następnie miasta pierwszego rodzica objęte tym zakresem są kopiowane bezpośrednio do dziecka (odpowiadające sobie pozycje objęte zakresem). W nastepnej kolejności miasta drugiego rodzica objęte zakresem, które jeszcze nie są uwzględnione w dziecku, są kopiowane do dziecka. W tym kroku pozycje wyznaczane są na podstawie poszukiwań pozycji, którą zajmuje odpowiadające (ta sama pozycja) miasto rodzica pierwszego w ścieżce rodzica drugiego. Taka procedura gwarantuje, że zostaną zapełnione miejsca, którym odpowiadały miasta już uwzględnione w ścieżce. Ostatecznie miasta rodzica drugiego nieobjęte zakresem są przepisywane na odpowiadące pozycje dziecka.

  Przykład: pierwszyRodzic: [0 1 **2 3** 4 0], drugiRodzic: [0 2 **3 1** 4 0], zakres(2,3)
  dziecko:
  1. Miasto poczatkowe zostaje przepisane- [**0** _ _ _ _ **0**]
  2. Geny pierwszego rodzica zostają przepisane- [0 _ **2 3** _ 0]
  3. Miasto w zakresie drugiego rodzica "1" nie wystepuje w dziecku. Miastu temu w pierwszym rodzicu odpowiada miasto "3". Miasto to w drugim rodzicu znajduje się w zakresie, a więc krok jest powtarzany. Miastu "3" w pierwszym rodzicu odpowiada miasto "2". Miasto to jest na pozycji 1 w drugim dziecku. Pozycja ta nie znajduje się w zakresie, a więc zostaje ona wybrana jako pozycja dla miasta "1".
  A więc: [0 **1** 2 3 _ 0]
  4. Pozostałe miasta uzupełniane są odpowiadającymi miastami w rodzicu drugim: [0 1 2 3 **4** 0]

1.3.5 Mutacja

Nowo powstała populacja złożona z dzieci oraz osobników, którzy nie zostali poddani krzyżowaniu (tak zadecydował rzut na prawdopodobieństwo krzyżowania), zostaje poddania mutacji, również z określonym prawdopodobieństwem. Zjawisko mutacji pozwala na zwiększenie dywersyfikacji. W wyniku mutacji mogą powstać pożądane cechy, których nie udałoby sie uzyskać za pomocą krzyżowania osobników.

- Mutacja Swap

  Losowane są dwie różne pozycje ścieżki, a następnie odpowiadające im miasta zostają wymienione między sobą. Przykład: [0 1 2 3 4 5 0] – Swap(1,4) –> [0 4 2 3 1 5 0]

- Mutacja Invert

  Losowany jest zakres pozycji ścieżki, a następnie występujące w tym zakresie miasta zostają odwrocone kolejnością. Przykład: [0 1 2 3 4 5 0] – Invert(1,4) –> [0 4 3 2 1 5 0]

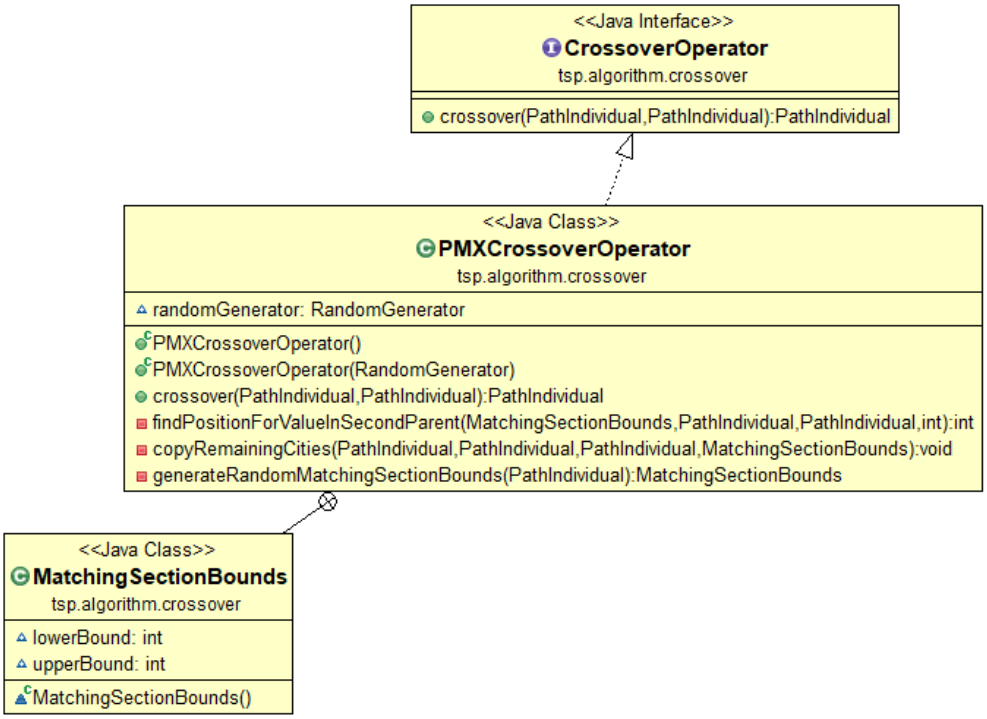
2 Implementacja algorytmu, opis istotnych klas

2.1 Instancja

Klasa reprezentująca instancję problemu. Przechowuje informacje o krawędziach w postaci macierzy sąsiedztwa.

2.2 CrossoverOperator

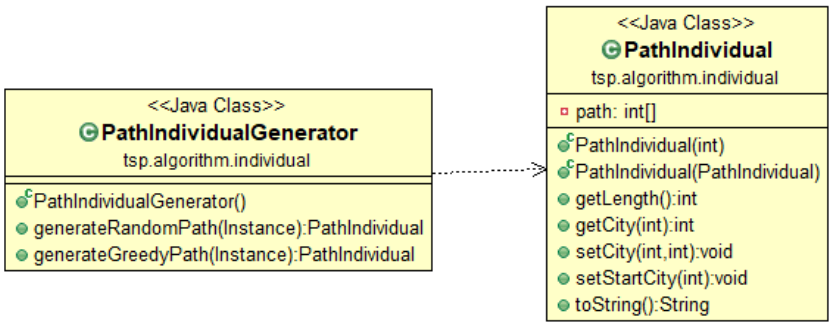
Interfejs CrossoverOperator deklaruję metodę crossover, która służy do krzyżowania dwóch osobników. Implementacja PMXCrossoverOperator realizuje metodę PMX opisaną w 1.3.4. Klasa pomocnicza MatchingSectionBounds służy do przechowywania wylosowanego zakresu dla krzyżowania.



Rysunek 1: Pakiet: crossover

2.3 PathIndividual

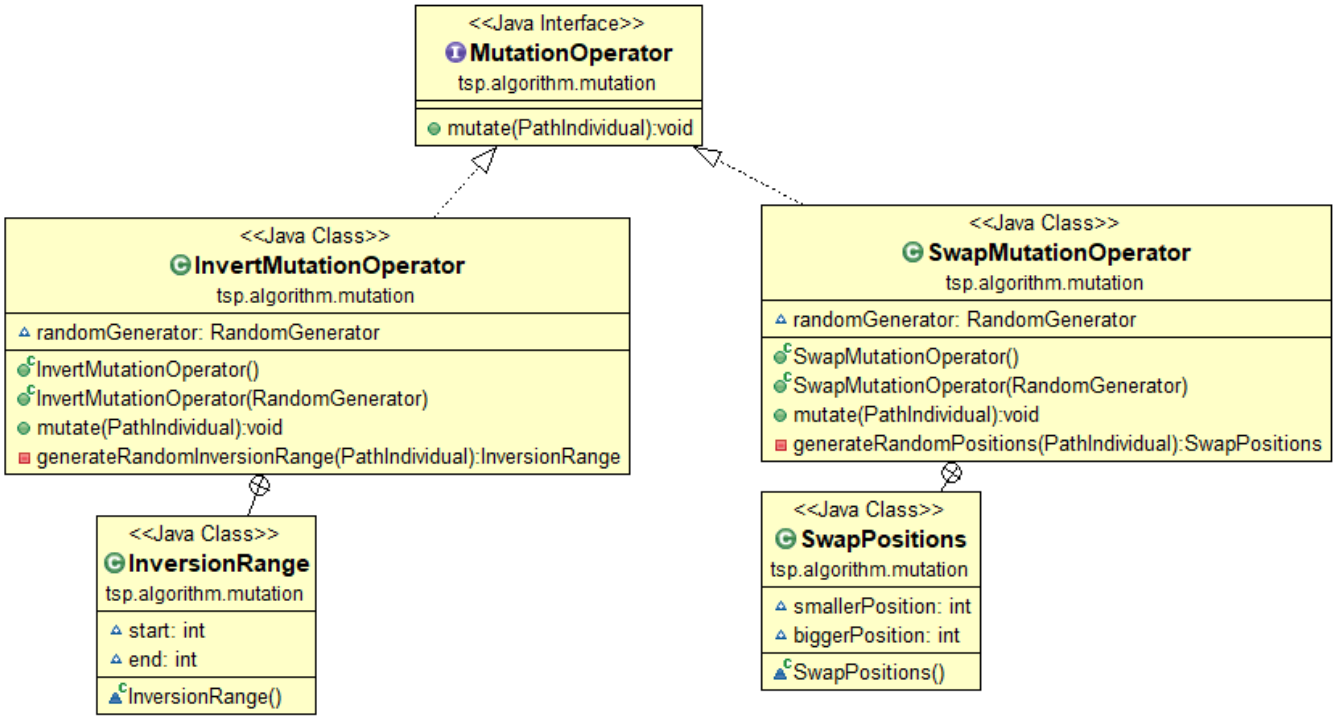
PathIndividual definiuje osobnika - rozwiązanie problemu komiwojażera. Klasa PathIndividualGenerator potrafi wygenerować losową ścieżkę, jak również, ścieżkę zachłanną.



Rysunek 2: Pakiet: individual

2.4 MutationOperator

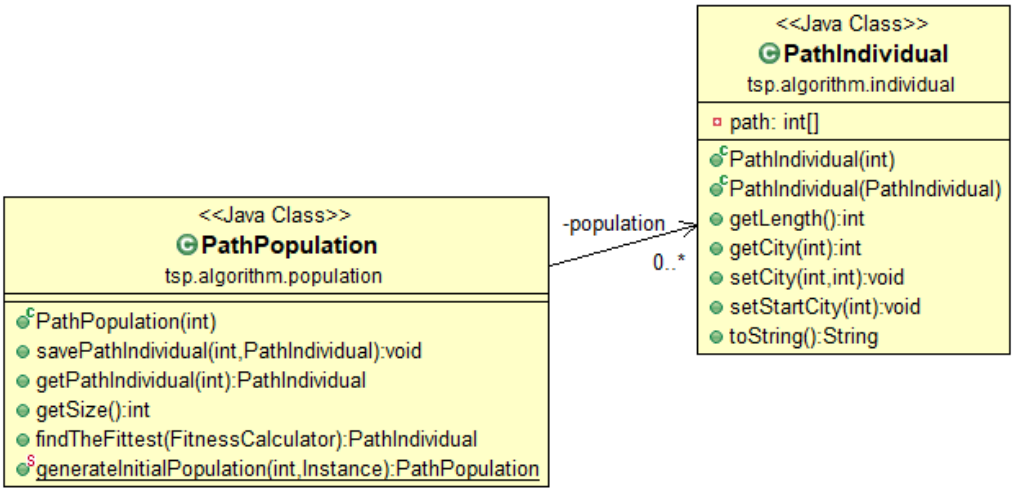
Interfejs MutationOperator jest implementowany przez dwa konkretne rodzaje mutacji - SwapMutationOperator oraz InvertMutationOperator. Zasada działania poszczególnych mutacji została opisana w podpunkcie 1.3.5.



Rysunek 3: Pakiet: mutation

2.5 PathPopulation

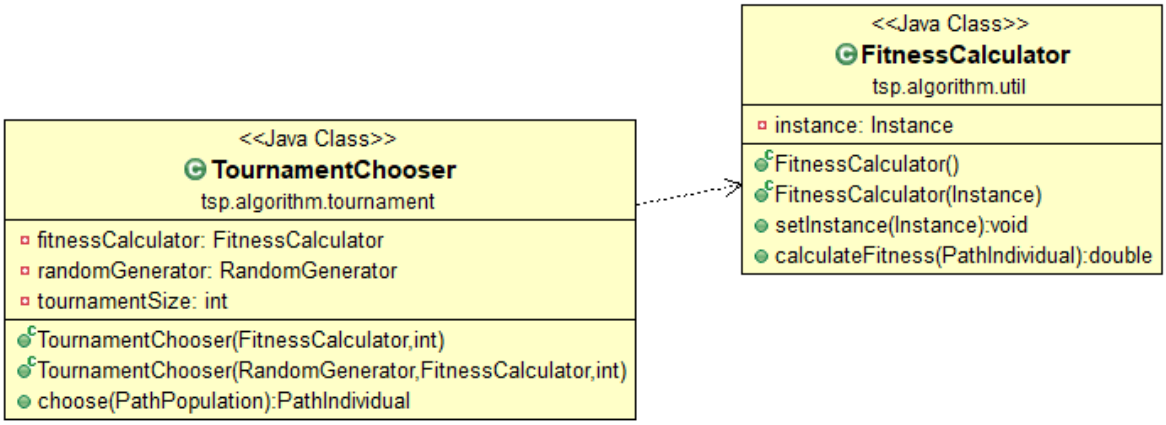
Klasa PathPopulation reprezentuje populację w algorytmie, a więc agreguje osobników (PathIndividual). Klasa deklaruje również operacje znalezienia najlepiej przystosowanego osobnika spośród wystepujących w populacji oraz metode statyczną generującą startową populację.



Rysunek 4: Pakiet: population

2.6 TournamentChooser, FitnessCalculator

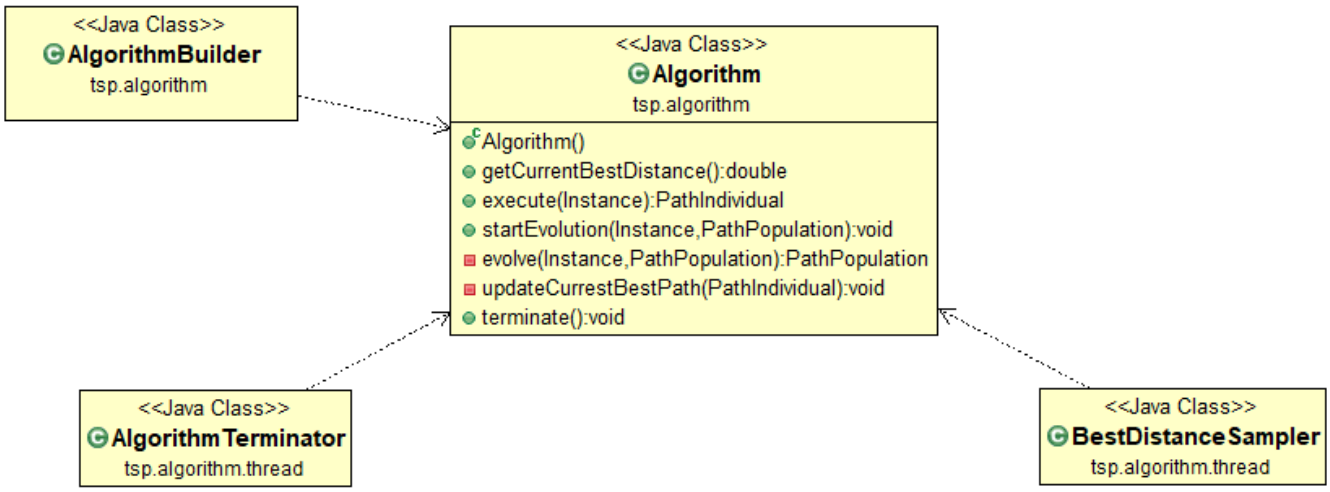
TournamentChooser służy do wyboru osobników z populacji metodą turniejową. FitnessCalculator udostępnia metodę calculateFitness służącą do wyznaczenia wartości przystosowania konkretnego osobnika.



Rysunek 5: Pakiety: tournament, util

2.7 Algorithm

Klasa Algorithm zarządza całym algorytmem (procesem ewolucji). AlgorithmBuilder jest klasą pomoczniczą, służącą do wygodnego tworzenia instancji klasy Algorithm podając jego konfigurację oraz inne parametry (wzorzec projektowy Builder). Klasa AlgorithmTerminator realizuje kryterium stopu w postaci czasu wykonywania się algorytmu. Klasa BestDistanceSampler zajmuje się próbkowaniem aktualnie najlepszej wartości ścieżki oraz zapisem jej do pliku / wypisaniem na ekran.



Rysunek 6: Pakiet: algorithm

3 Wyniki badań

Po zaimplementowaniu algorytmu oraz klas pomoczniczych zostały przeprowadzone testy jakościowe.

3.1 Parametry

- Badania przeprowadzono na instancjach eil101.tsp, rbg403.atsp oraz d1291.tsp
- Algorytm został zbadany dla następujących wielkości populacji: 1/10*N, 1/2*N, N
  (gdzie N jest wielkością instancji)
- Dla powyższych wielkości populacji zostały zbadane mutacje Swap oraz Invert.
- Dla najlepszej wielkości populacji oraz rodzaju mutacji został zbadany wpływ współczynnika mutacji. Wartości: 0.02, 0.05, 0.1
- Ograniczenie czasowe algorytmu zostało ustawione na 6 minut.

3.2 Instancja - eil101.tsp

Optymalna długość ścieżki: 629
Najlepsza znaleziona długość ścieżki: 675

3.2.1 Tabele zbiorcze

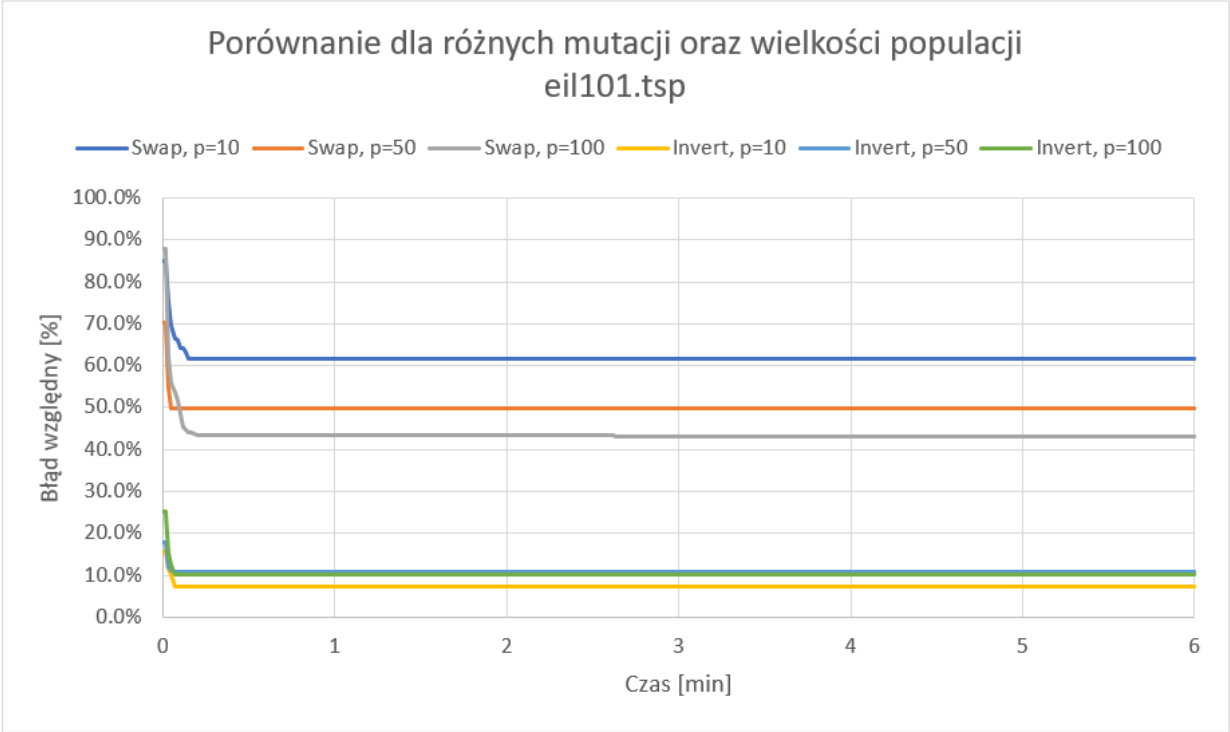
Wielkosć populacji		10		50		100	
Rodzaj ruchu		Swap	Invert	Swap	Invert	Swap	Invert
Czas [min]	0	84.90%	15.74%	49.76%	17.97%	43.40%	25.12%
	0.5	61.53%	7.31%	70.27%	10.97%	87.76%	10.33%
	1	61.53%	7.31%	49.76%	10.97%	43.40%	10.33%
	1.5	61.53%	7.31%	49.76%	10.97%	43.40%	10.33%
	2	61.53%	7.31%	49.76%	10.97%	43.40%	10.33%
	2.5	61.53%	7.31%	49.76%	10.97%	43.40%	10.33%
	3	61.53%	7.31%	49.76%	10.97%	43.24%	10.33%
	3.5	61.53%	7.31%	49.76%	10.97%	43.24%	10.33%
	4	61.53%	7.31%	49.76%	10.97%	43.24%	10.33%
	4.5	61.53%	7.31%	49.76%	10.97%	43.24%	10.33%
	5	61.53%	7.31%	49.76%	10.97%	43.24%	10.33%
	5.5	61.53%	7.31%	49.76%	10.97%	43.24%	10.33%
	6	61.53%	7.31%	49.76%	10.97%	42.24%	10.33%

Rysunek 7: Zależnosc błędu względnego od czasu dla różnych kombinacji parametrów (eil101.tsp)

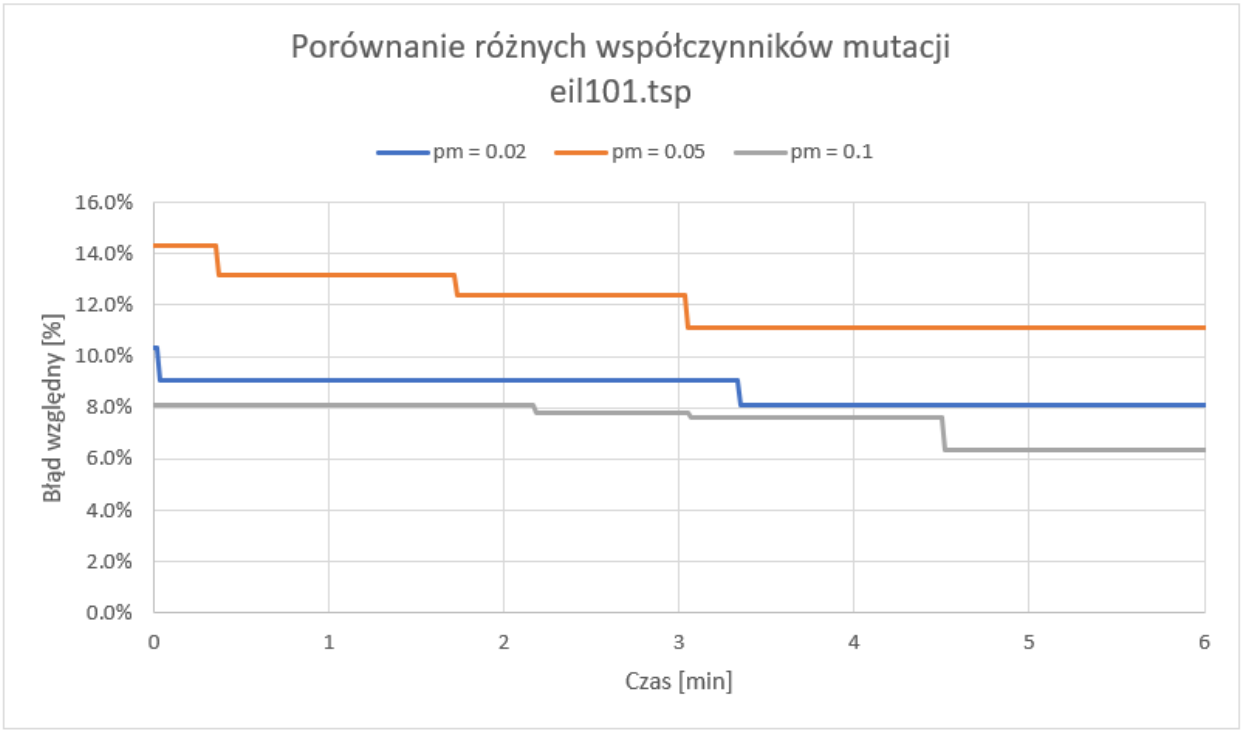
Wielkosć populacji		10		
Rodzaj ruchu		Invert		
Współczynnik mutacji		0.02	0.05	0.1
Czas [min]	0	10.33%	14.31%	8.11%
	0.5	9.06%	13.20%	8.11%
	1	9.06%	13.20%	8.11%
	1.5	9.06%	13.20%	8.11%
	2	9.06%	12.40%	8.11%
	2.5	9.06%	12.40%	7.79%
	3	9.06%	12.40%	7.79%
	3.5	8.11%	11.13%	7.63%
	4	8.11%	11.13%	7.63%
	4.5	8.11%	11.13%	7.63%
	5	8.11%	11.13%	6.36%
	5.5	8.11%	11.13%	6.36%
	6	8.11%	11.13%	6.36%

Rysunek 8: Zależnosc błędu względnego od czasu dla różnych współczynników mutacji oraz najlepszej kombinacji z poprzedniej tabeli (eil101.tsp)

3.2.2 Wykresy



Rysunek 9: Zależnośc błędu względnego od czasu dla różnych kombinacji parametrów (eil101.tsp)



Rysunek 10: Zależnośc błędu względnego od czasu dla różnych współczynników mutacji - pm (eil101.tsp)

3.3 Instancja - rbg403.atsp

Optymalna długość ścieżki: 2465
Najlepsza znaleziona długość ścieżki: 2755

3.3.1 Tabele zbiorcze

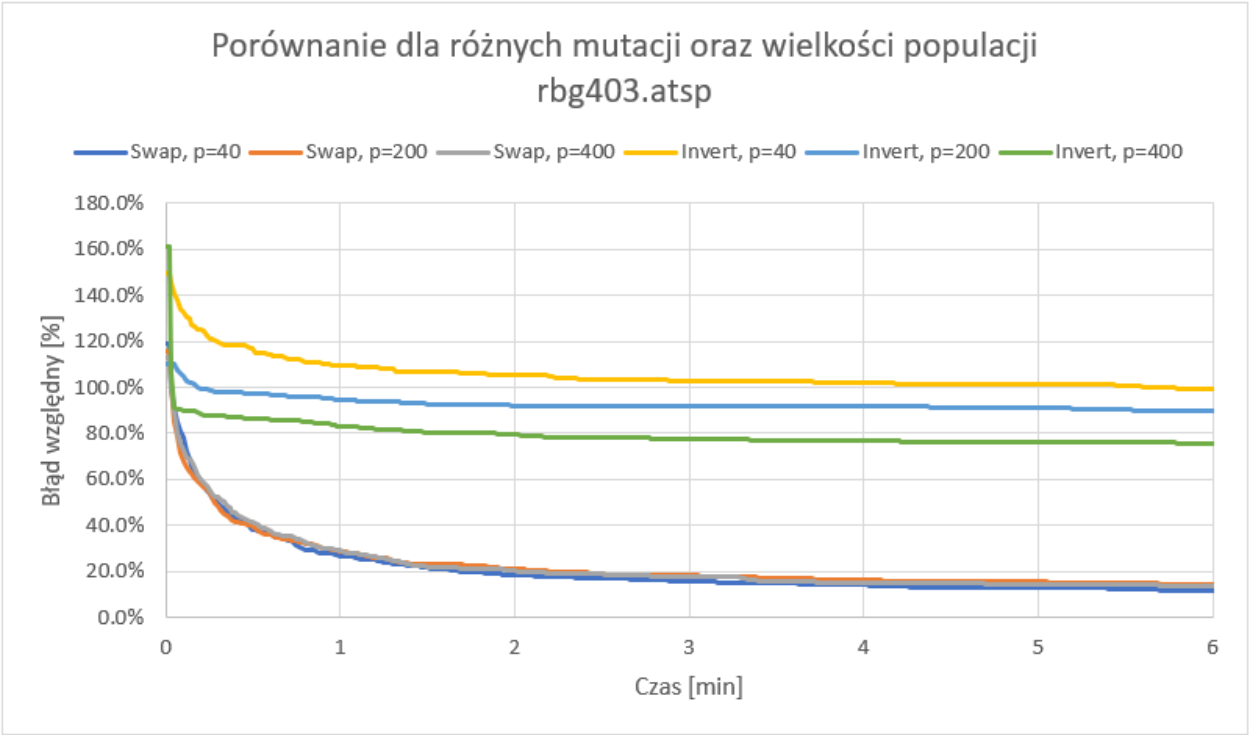
Wielkosć populacji		40		200		400	
Rodzaj ruchu		Swap	Invert	Swap	Invert	Swap	Invert
Czas [min]	0	118.90%	149.29%	115.66%	110.39%	112.82%	161.05%
	0.5	38.34%	116.63%	39.43%	97.40%	41.54%	86.57%
	1	26.86%	109.37%	28.76%	94.69%	29.21%	83.33%
	1.5	21.66%	106.61%	23.37%	92.62%	22.03%	80.65%
	2	18.58%	105.72%	21.14%	91.93%	20.32%	79.55%
	2.5	17.04%	103.61%	19.07%	91.89%	19.19%	78.01%
	3	15.74%	103.00%	18.30%	91.60%	18.13%	77.77%
	3.5	15.21%	102.43%	17.36%	91.56%	16.11%	77.20%
	4	14.24%	102.07%	16.23%	91.56%	15.50%	77.16%
	4.5	13.35%	101.34%	16.11%	91.44%	14.97%	76.39%
	5	13.27%	101.14%	15.58%	91.24%	14.77%	76.35%
	5.5	12.58%	100.89%	15.25%	90.26%	14.44%	75.98%
	6	11.76%	99.35%	14.28%	89.66%	13.59%	75.82%

Rysunek 11: Zależnosc błędu względnego od czasu dla różnych kombinacji parametrów (rbg403.atsp)

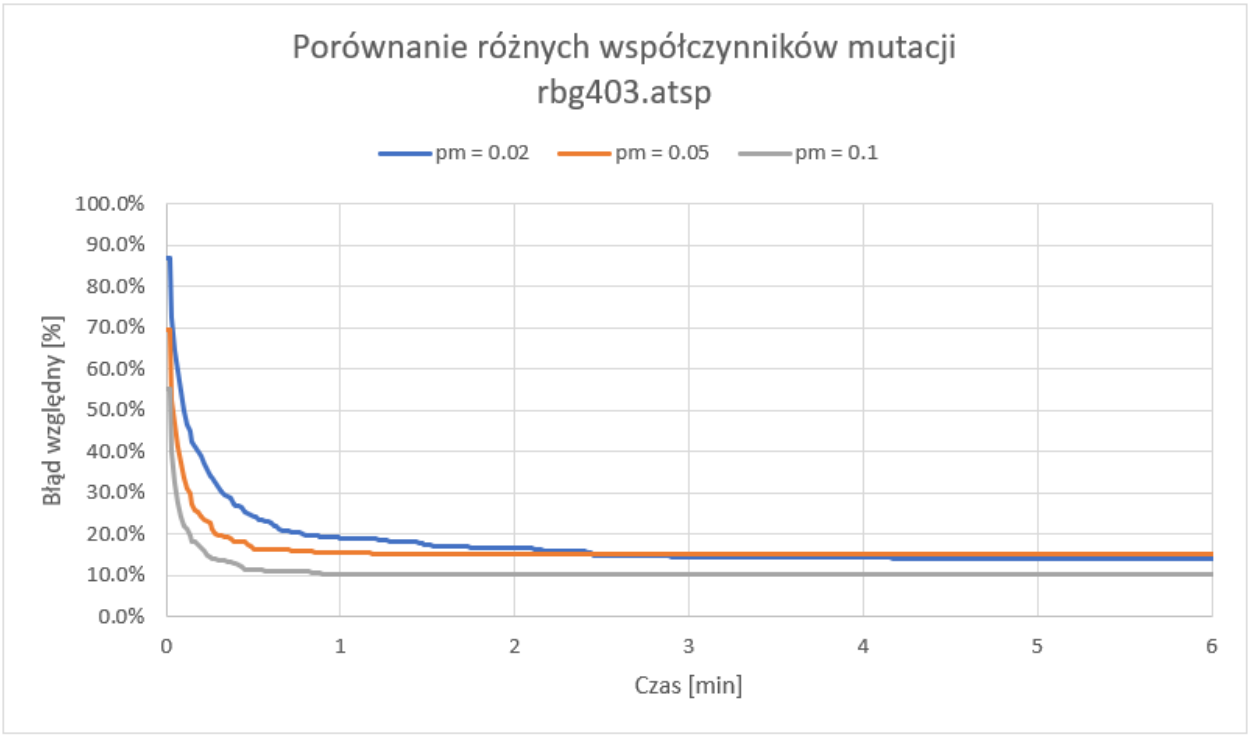
Wielkosć populacji		40		
Rodzaj ruchu		Swap		
Współczynnik mutacji		0.02	0.05	0.1
Czas [min]	0	86.69%	69.61%	55.05%
	0.5	24.42%	16.51%	11.28%
	1	19.11%	15.74%	10.34%
	1.5	17.32%	15.21%	10.34%
	2	16.80%	15.21%	10.34%
	2.5	14.97%	15.21%	10.34%
	3	14.65%	15.21%	10.34%
	3.5	14.60%	15.21%	10.34%
	4	14.28%	15.21%	10.30%
	4.5	14.24%	15.21%	10.30%
	5	14.08%	15.21%	10.14%
	5.5	13.91%	15.21%	10.14%
	6	13.91%	15.21%	10.14%

Rysunek 12: Zależnosc błędu względnego od czasu dla różnych współczynników mutacji oraz najlepszej kombinacji z poprzedniej tabeli (rbg403.atsp)

3.3.2 Wykresy



Rysunek 13: Zależnośc błędu względnego od czasu dla różnych kombinacji parametrów (rbg403.atsp)



Rysunek 14: Zależnośc błędu względnego od czasu dla różnych współczynników mutacji - pm (rbg403.atsp)

3.4 Instancja - d1291.tsp

Optymalna długość ścieżki: 50801
Najlepsza znaleziona długość ścieżki: 234287

3.4.1 Tabele zbiorcze

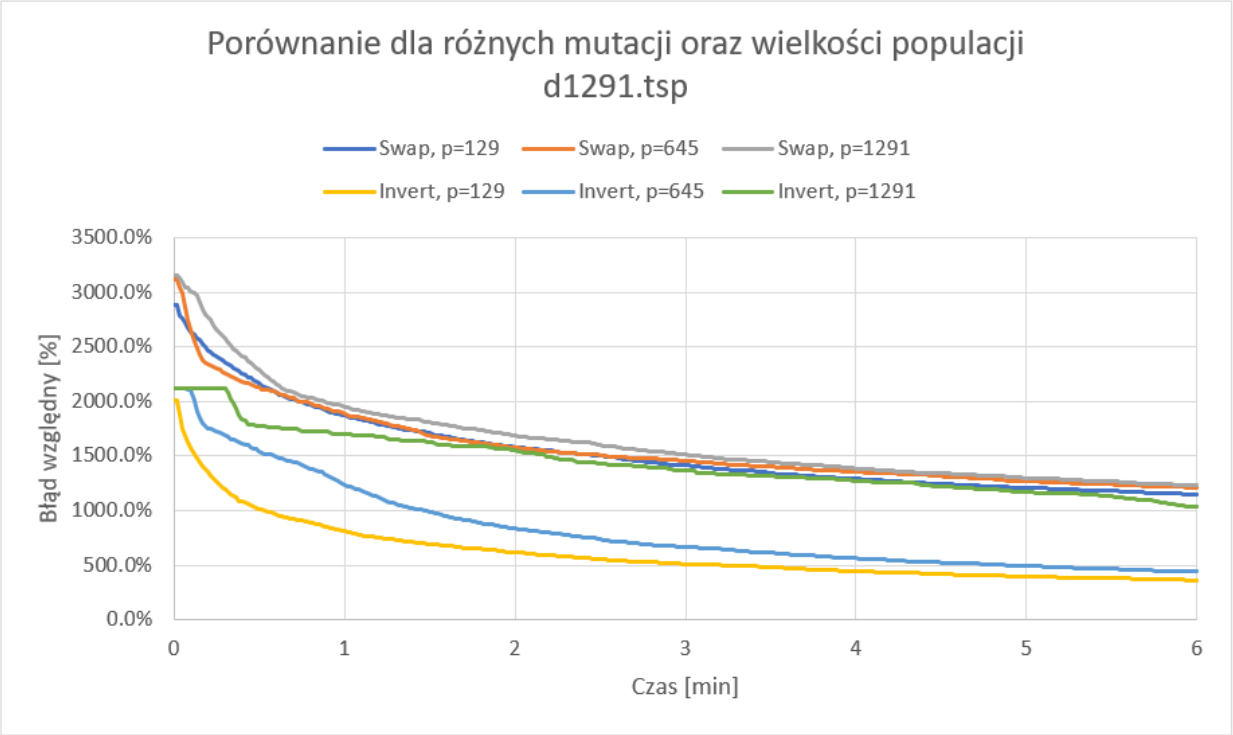
Wielkosć populacji		129		645		1291	
Rodzaj ruchu		Swap	Invert	Swap	Invert	Swap	Invert
Czas [min]	0	2889%	2011%	3117%	2117%	3154%	2117%
	0.5	2171%	1016%	2134%	1543%	2291%	1777%
	1	1875%	812%	1886%	1235%	1955%	1699%
	1.5	1717%	694%	1686%	993%	1811%	1625%
	2	1584%	618%	1579%	835%	1690%	1550%
	2.5	1502%	555%	1511%	739%	1603%	1434%
	3	1414%	512%	1459%	668%	1512%	1369%
	3.5	1345%	481%	1401%	612%	1447%	1317%
	4	1290%	446%	1352%	562%	1387%	1267%
	4.5	1247%	418%	1316%	525%	1342%	1223%
	5	1209%	396%	1267%	494%	1299%	1171%
	5.5	1180%	380%	1241%	466%	1267%	1132%
	6	1148%	361%	1211%	440%	1232%	1029%

Rysunek 15: Zależnosc błędu względnego od czasu dla różnych kombinacji parametrów (d1291.tsp)

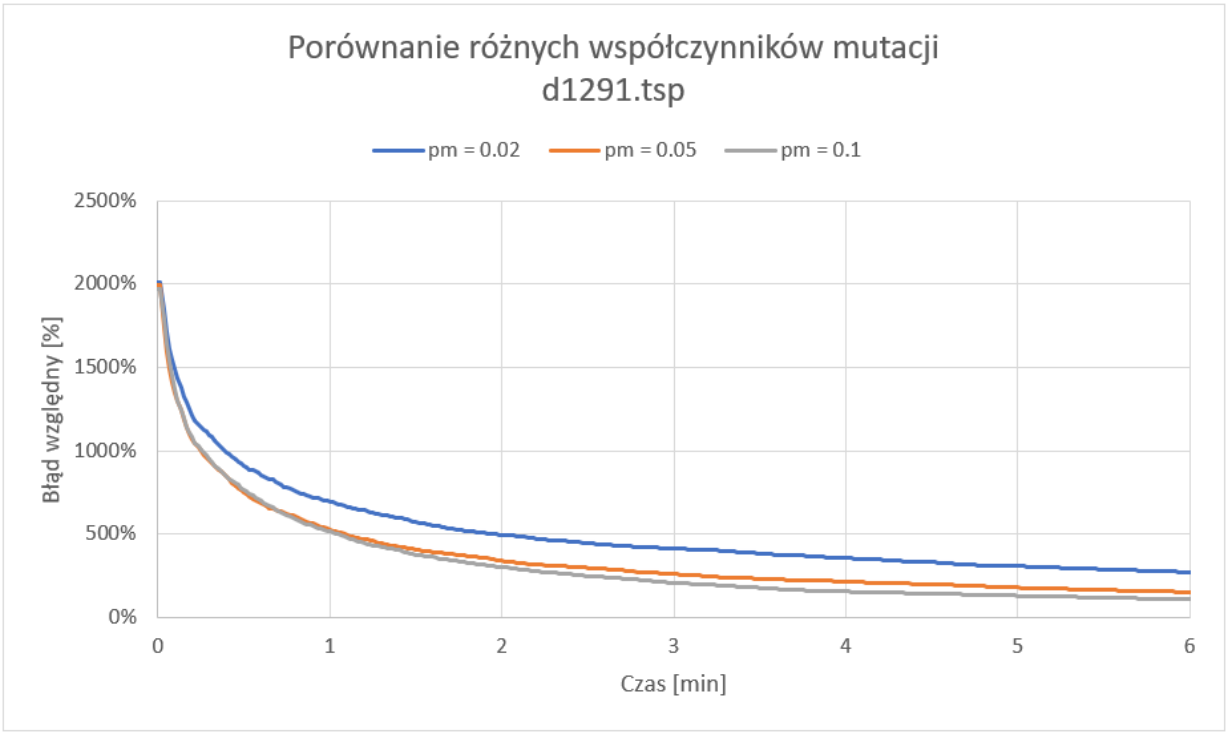
Wielkosć populacji		129		
Rodzaj ruchu		Invert		
Współczynnik mutacji		0.02	0.05	0.1
Czas [min]	0	2008%	1991%	1973%
	0.5	910%	750%	765%
	1	695%	527%	516%
	1.5	570%	407%	377%
	2	495%	338%	300%
	2.5	446%	297%	247%
	3	410%	259%	208%
	3.5	384%	231%	177%
	4	356%	215%	155%
	4.5	331%	198%	141%
	5	307%	178%	130%
	5.5	287%	164%	118%
	6	271%	150%	108%

Rysunek 16: Zależnosc błędu względnego od czasu dla różnych współczynników mutacji oraz najlepszej kombinacji z poprzedniej tabeli (d1291.tsp)

3.4.2 Wykresy



Rysunek 17: Zależnośc błędu względnego od czasu dla różnych kombinacji parametrów (d1291.tsp)



Rysunek 18: Zależnośc błędu względnego od czasu dla różnych współczynników mutacji - pm (d1291.tsp)

3.5 Porównanie z algorytmem opartym na Tabu Search

3.5.1 Instancja - eil101.tsp

Najmniejszy błąd względny uzyskany metodą tabu search: 0.00%
Najmniejszy błąd względny uzyskany metodą ewolucyjną: 6.36%

3.5.2 Instancja - rbg403.atsp

Najmniejszy błąd względny uzyskany metodą tabu search: 1.34%
Najmniejszy błąd względny uzyskany metodą ewolucyjną: 10.14%

3.5.3 Instancja - d1291.tsp

Najmniejszy błąd względny uzyskany metodą tabu search: 115%
Najmniejszy błąd względny uzyskany metodą ewolucyjną: 108%

4 Podsumowanie, wnioski

Zaimplementowany algorytm skutecznie radził sobie ze znajdywaniem rozwiązań bliskich optymalnemu w krótkim czasie, mimo dużych rozmiarów instancji.

- Algorytm jest bardzo wrażliwy na konfigurację. Dobór parametrów do konkretnej instancji problemu jest kluczowy w celu uzyskania dobrych wyników.
- Dla instancji średniej wielkości lepiej sprawdził się algorytm oparty na poszukiwaniu z zakazami. Jednak dla instancji dużej (1291) wynik bliższy optymalnemu w tym samym czasie znalazł algorytm ewolucyjny.
- Słabsze wyniki dla średnich instancji mogą wynikać ze złego dopasowania konfiguracji parametrów dla algorytmu genetycznego.
- Mutacja typu Invert sprawdza się lepiej dla problemu symetrycznego.
- Mutacja typu Swap sprawdza się lepiej dla problemu asymetrycznego.
- Dla każdej z instancji ustawienie parametru mutacji na najwyższą badaną wartośc (0.1) dało najlepsze rezultaty.
- Najszybszy wzrost jakości rozwiązań występuje w początkowej fazie działania algorytmu genetycznego (duże nachylenie w początkowej części wykresów). Oznacza to najszybszy rozwój populacji w poczatkowym stadium ewolucji.
- W związku z powyższym algorytm jest w stanie znaleźć zadowalające rozwiązanie w bardzo szybkim czasie.
- Okres szybkiego wzrostu jakości rozwiązań jest proporcjonalny do wielkości instancji. Dla instancji wielkości 101 trwał niecałe 10s, dla instancji wielkości 1291- minutę.
- Na wzrost jakości rozwiązań może mieć duży wpływ skonfigurowana dywersyfikacja, która w przypadku tego algorytmu może być zwiększona za pomocą parametru mutacji oraz wielkości turniejów.
- Im mniejsze turnieje tym większa dywersyfikacja - dopuszczanych jest więcej słabszych rozwiązań, które dla dużych turniejów są wypierane przez rozwiązania najlepsze w populacji. Dywersyfikacja może prowadzić do lepszych rezultatów końcowych.
- Dla badanych instancji oraz pozostałych parametrów, najlepsze wyniki okazało się generować ustawienie najmniejszej wielkości populacji spośród badanych a więc 1/10*N (N- wielkość instancji).