Algorytmy metaheurystyczne 3

Paweł Cegieła, Wojciech Sęk

26 maja 2022

1 Rodzaje algorytmu

Algorytm genetyczny można podzielić ze względu na dwie znaczące modyfikacje:

- algorytm może działać na jednej populacji lub na wielu populacjach żyjących na osobnych wyspach
- algorytm może działać całkowicie sekwencyjnie lub wykonywać część obliczeń równolegle

2 Parametry algorytmów

Dla algorytmu sekwencyjnego z jedną populacją podajemy następujące parametry:

- *matrix*: macierz przechowująca odległości między miastami z problemu TSP
- *gen_rand*: wartość logiczna mówiąca, czy pierwsze pokolenie ma być całkowicie losowe czy przybliżone innymi metaheurystykami
- *gen_size*: liczność populacji na każdym etapie algorytmu
- *elite_num*: liczność elity, czyli najlepszych rozwiązań, które przeżywają i przechodzą do następnego pokolenia
- *cross_op*: wartość enumeratywna mówiąca o tym, jaki typ krzyżowania zostanie użyty w algorytmie. Możliwe wartości:
  - 0 - HALF CROSSOVER: dziecko w każdym kroku generowania przyjmuje kolejne od jednego z rodziców miasto z prawdopodobieństwem 1/2, bierzemy najwcześniejsze miasto z rodzica, które nie występuje w dziecku
  - 1 - ORDER CROSSOVER: pół dziecka to podciąg rodzica, drugie pół to podciąg drugiego rodzica składający się z nieużytych wcześniej miast i zaczynający od miasta występującego na tym miejscu w pierwszym rodzicu
  - 2 - CYCLE CROSSOVER: bierzemy miasto z pierwszego rodzica, patrzymy na miasto pod danym indeksem w drugim rodzicu, idziemy do niego w pierwszym rodzicu, powtarzamy aż zamkniemy cykl. Resztę uzupełniamy drugim rodzicem.
  - 3 - PARTIALLY MAPPED CROSSOVER: na danym podciągu indeksów tworzymy mapę między rodzicami, następnie używamy pociągu z jednego rodzica i zmapowanych wartości z drugiego.
- *swap_change*: wartość logiczna decydująca o tym, czy w trakcie mutacji poruszamy się w sąsiedztwie *swap* czy *reverse*
- *size_of_tournament*: rozmiar turnieju przy losowaniu rodziców do tworzenia nowego pokolenia. Wartość ustawiona na 0 sprawia, że zamiast turnieju korzystamy z zasady ruletki, gdzie funkcja dopasowania jest odwrotnością wagi danej permutacji.
- *mut_chance*: prawdopodobieństwo, że dany osobnik zmutuje.
- *max_time*: ograniczenie czasowe, po którym algorytm kończy działanie. Wynik zwracany przez algorytm to najlepszy wynik znaleziony przed przekroczeniem limitu czasowego.

Dla algorytmu wyspowego podajemy także:

- *isles_num*: liczba wysp
- *migration_freq*: liczba iteracji, które dzielą od siebie moment wymiany genetycznej między wyspami

W algorytmach urównoleglonych podaje się również parametr *num_of_threads*, który mówi o liczbie wątków.

3 Teoretyczna złożoność

Rozważmy złożoności poszczególnych etapów w algorytmie sekwencyjnym bez wysp. Niech n to liczba miast, a k to liczność pokolenia:

- **Wyznaczanie populacji początkowej**: w języku Rust losowanie permutacji ma złożoność $O(1)$, zatem losowa generacja osobników ma złożoność $O(k)$. Dla przybliżeń przez inne metaheurystyki co najwyżej $\frac{k}{4}$ jest przybliżone, w naszym przypadku dajemy jeden raz $2 - OPT$ ze złożonością $O(n^3)$ i pozostałe to $nearest - neighbours$ od losowego miasta ze złożonością $O(n^2)$.
- **Ewaluacja**: Obliczanie wartości danej permutacji ma złożoność $O(n)$, robimy to k razy, zatem cały krok ma $O(kn)$.
- **Selekcja**: Wybieramy k rodziców. Przy ruletce wybór rodzica ma złożoność $O(1)$, przy turnieju $O(l)$, gdzie l to rozmiar turnieju. Zatem $O(k)$ dla ruletki i $O(kl)$ dla turnieju.
- **Krzyżowanie**: Każde z krzyżowań odbywa się w czasie liniowym względem długości permutacji, czyli $O(n)$, przy tworzeniu całej permutacji mamy $O(kn)$.
- **Mutacja**: mutacja typu *swap* ma złożoność $\Theta(1)$, a typu *reverse* $\Theta(n)$. Mutacji wykonujemy $\Theta(pn)$, gdzie p to prawdopodobieństwo mutacji. Zatem ta faza ma złożoność $\Theta(pn)$ lub $\Theta(pn^2)$. W worst case możemu wylosować wszystkie, więc odpowiednio złożoności są $O(n)$ i $O(n^2)$.

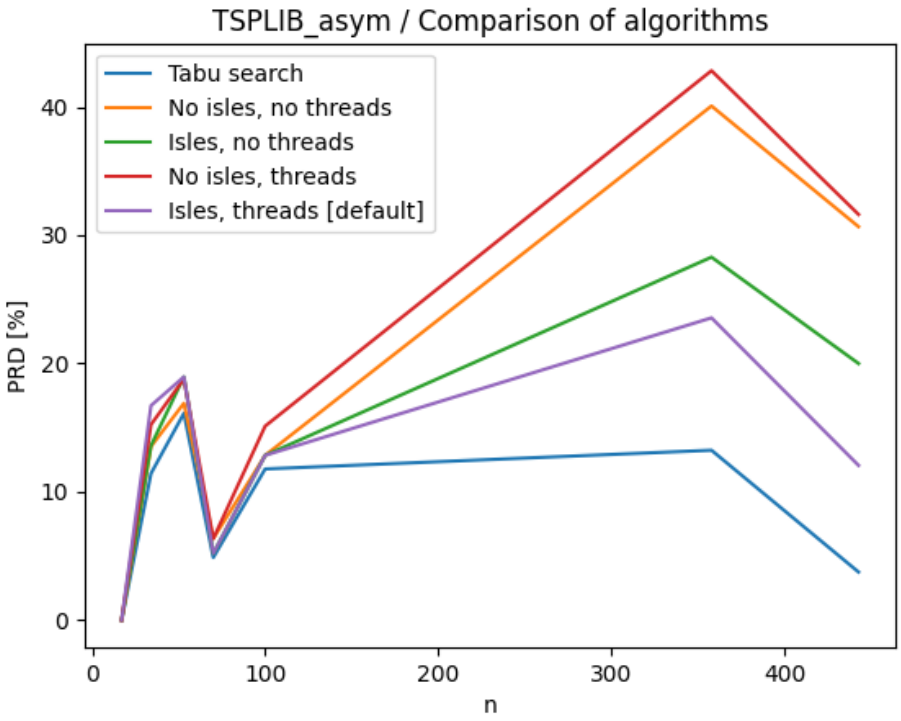
Ponadto w algorytmie wyspowym regularnie w niektórych iteracjach wymieniana jest informacja genetyczna ze złożonością $O(k^2)$ (każda wyspa wymienia się z każdą jakimś osobnikiem).
Podsumowując, w najgorszym przypadku:

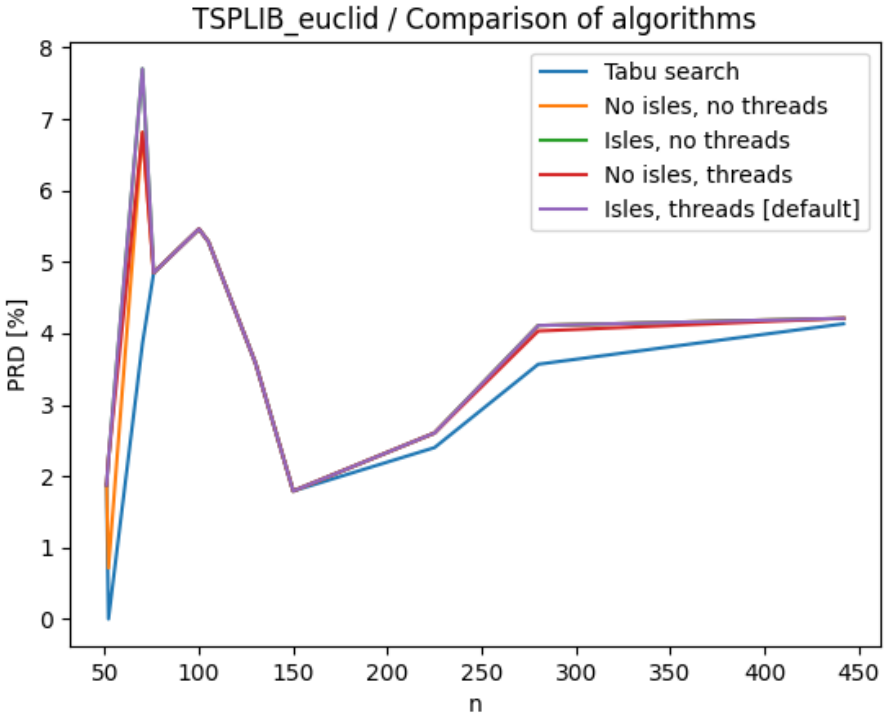
- **Generacja początkowej populacji**:
$$O(n^3 + kn^2 + k) = O((n + k)n^2)$$
- **Właściwa iteracja:**
$$O(kn) + O(kl) + O(kn) + O(n^2) + O(k^2) = O(k(n + l + k) + n^2)$$

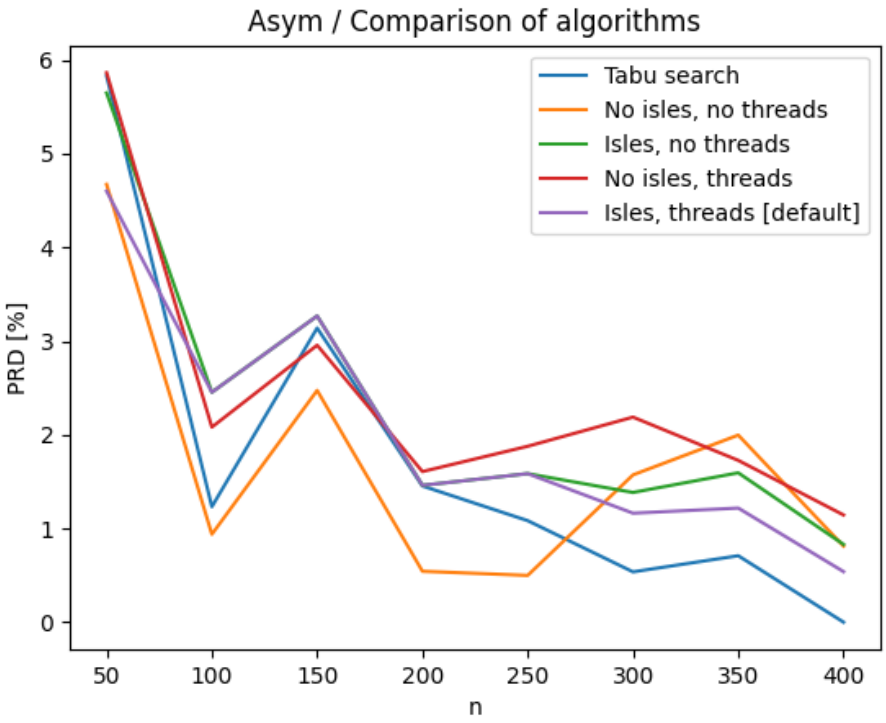
4 Wykonane eksperymenty

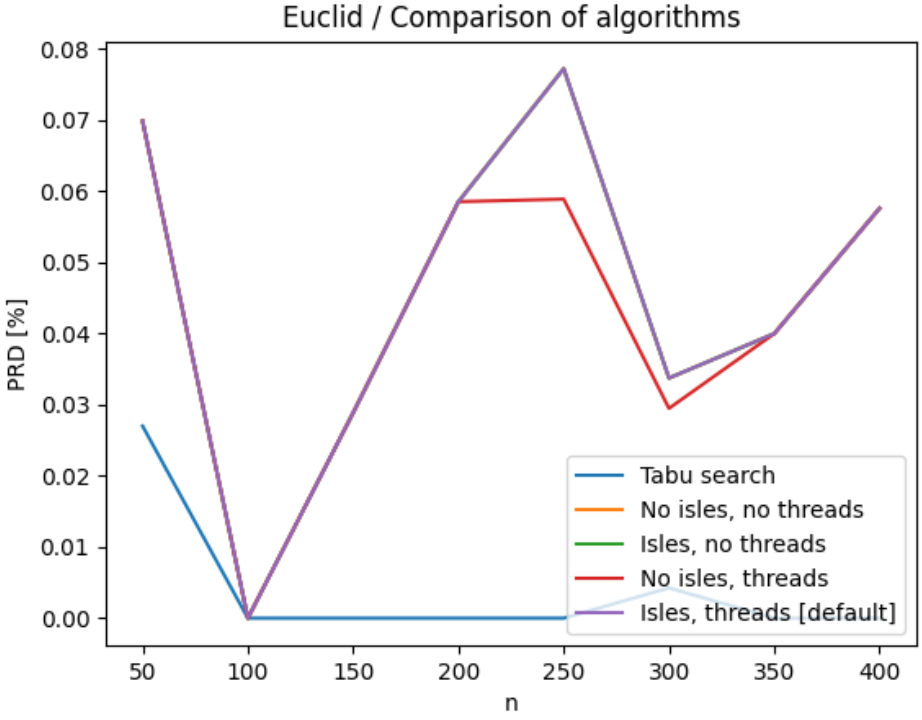
Strojenie algorytmu zostało wykonane na podstawie danych z TSPLIB. Wybrane zostały parametry, dla których algorytm najlepiej zachowywał się podczas testowania. Jeśli jeden z dwóch parametrów miał np. lepsze wyniki od drugiego dla jednych danych (np. dla mniejszych wielkości macierzy), a gorsze dla innych, wybierany był jeden z tych parametrów po przemyśleniu.
Po strojeniu zostały przeprowadzone eksperymenty porównujące algorytm w wersji oraz z parametrami wybranymi jako domyślne: z innymi wersjami algorytmu oraz z tabu search, a także z różnymi wartościami parametrów. Nie w każdym eksperymencie czy nie dla każdego typu macierzy domyślny algorytm okazał się najlepszy, natomiast strojenie pomogło nam znaleźć taki algorytm oraz parametry, które średnio zachowują się najlepiej.

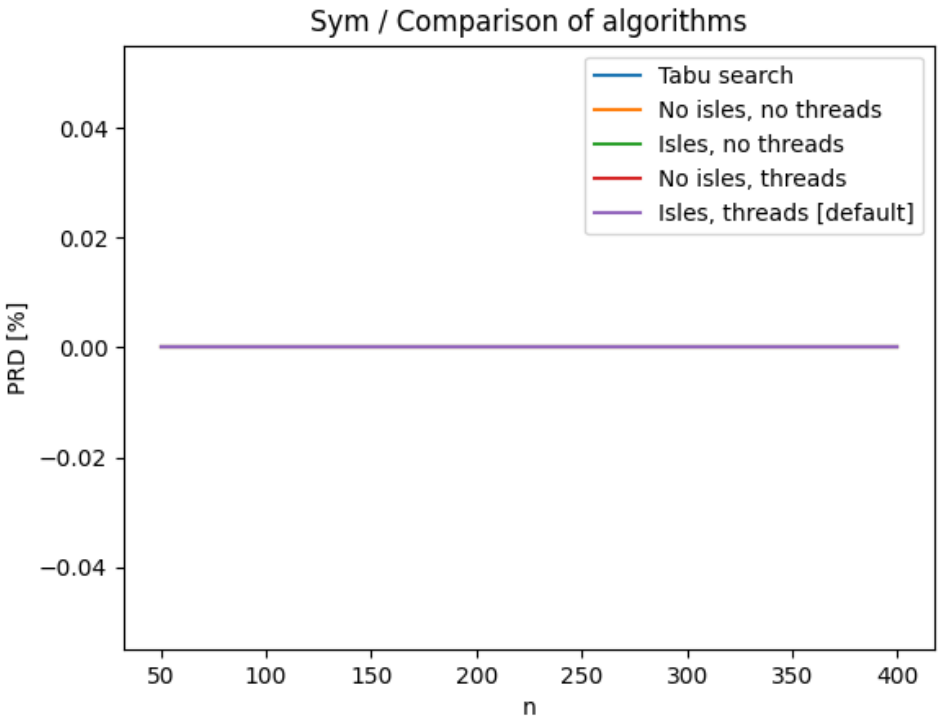
4.1 Porównanie wersji algorytmu genetycznego i tabu search



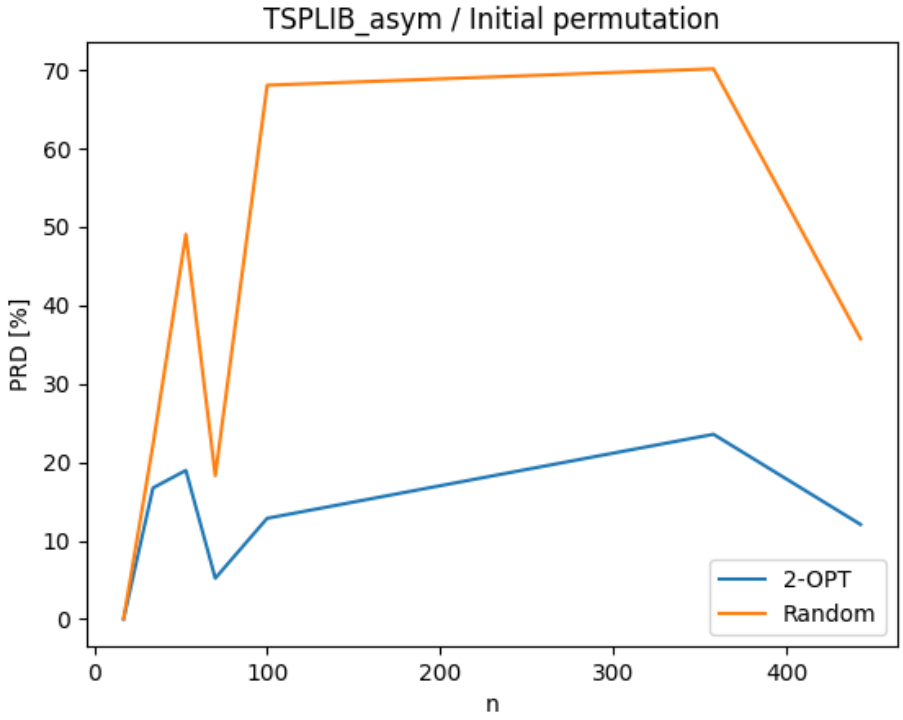


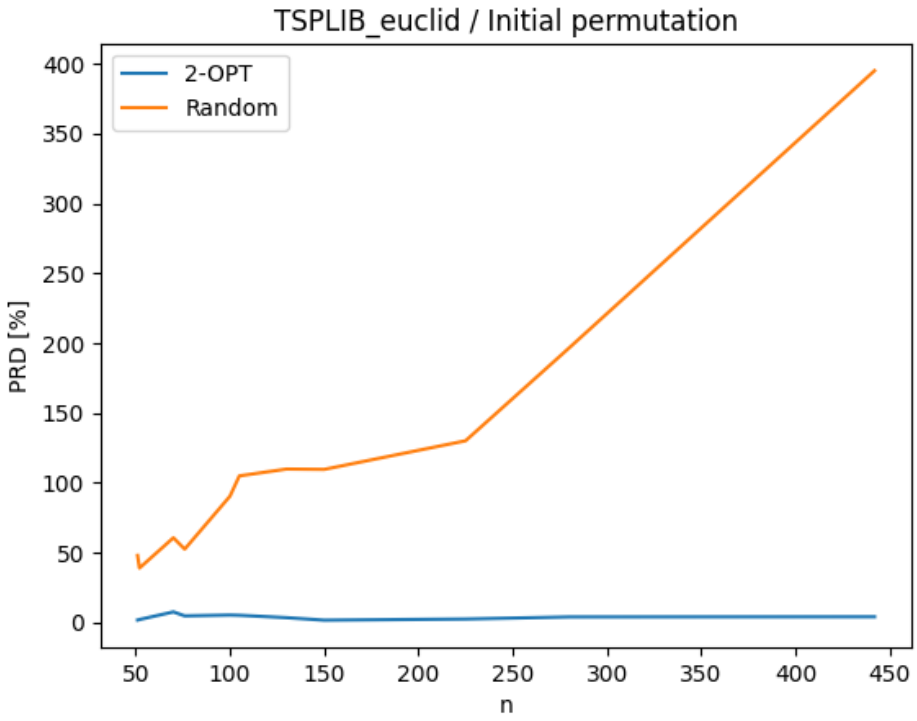

Asym / Comparison of algorithms
PRD [%]
n
Tabu search
No isles, no threads
Isles, no threads
No isles, threads
Isles, threads [default]
0
1
2
3
4
5
6
50
100
150
200
250
300
350
400

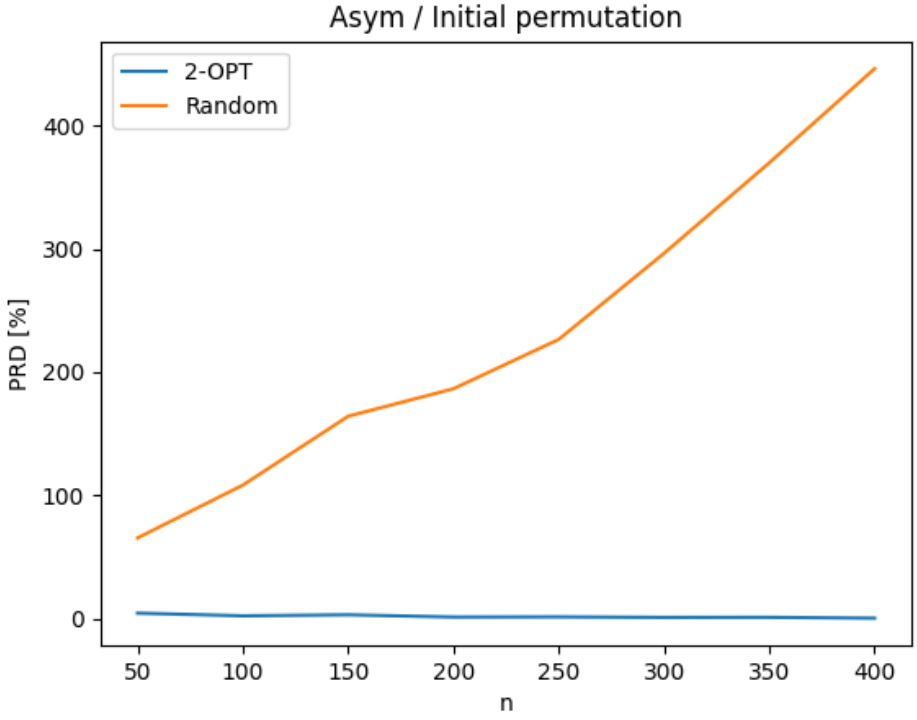

Euclid / Comparison of algorithms
PRD [%]
n
Tabu search
No isles, no threads
Isles, no threads
No isles, threads
Isles, threads [default]
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
50
100
150
200
250
300
350
400

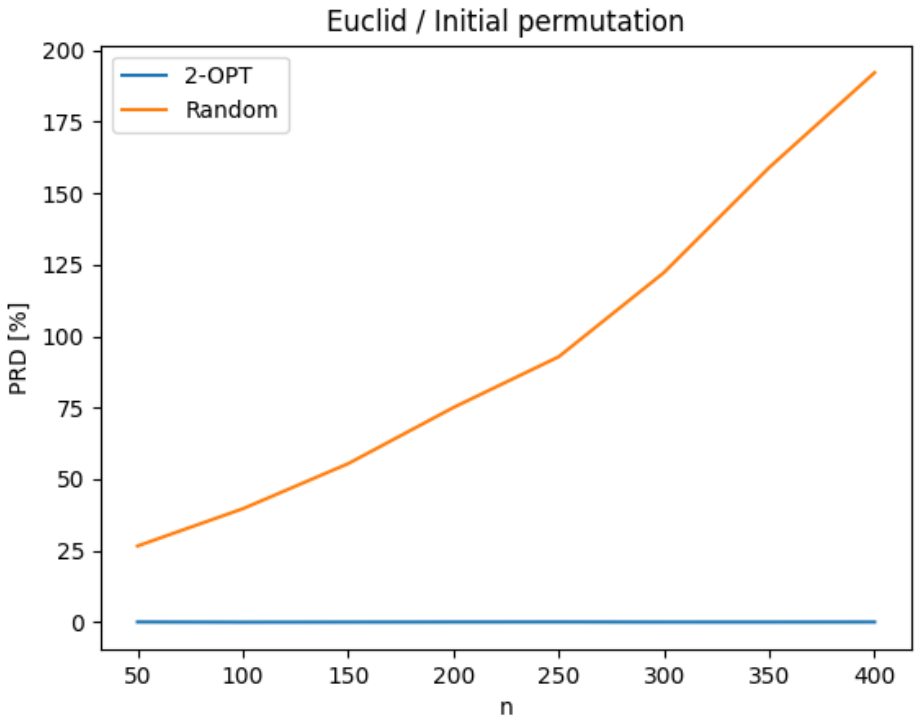


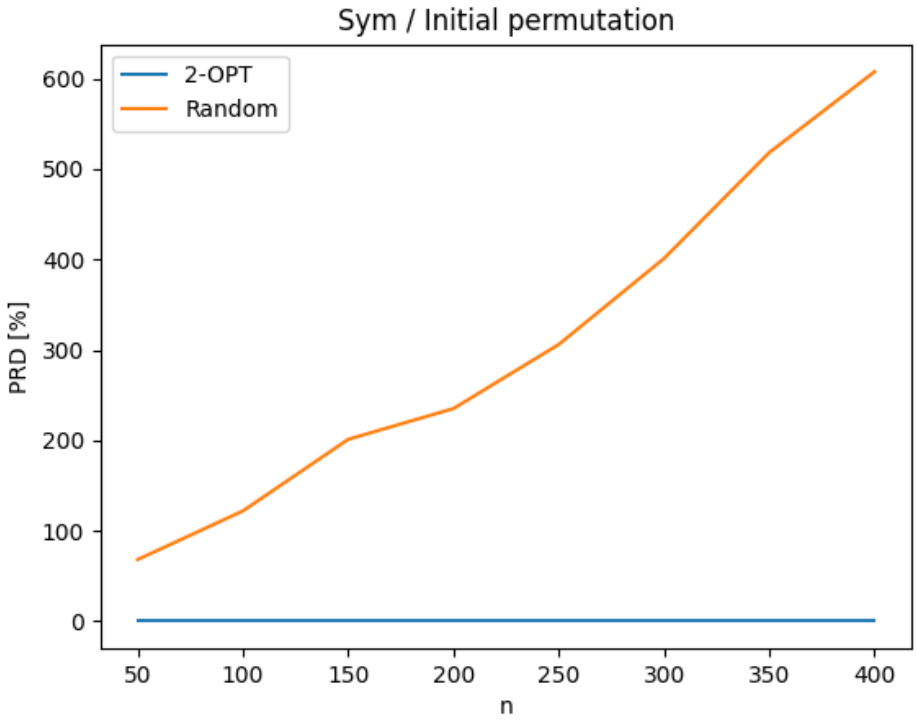
4.2 Przybliżenie początkowe



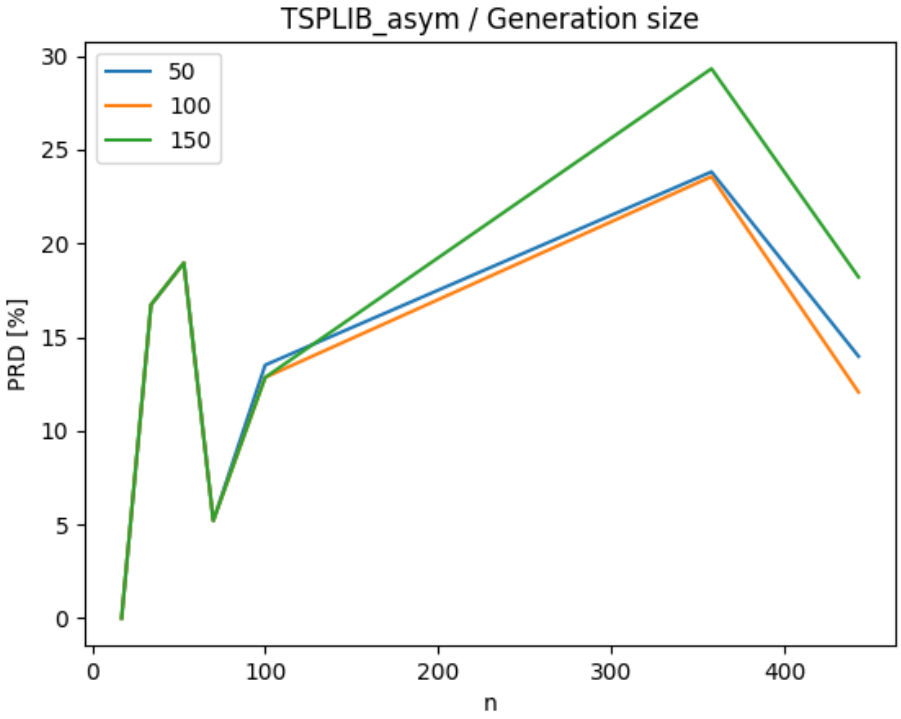

TSPLIB_euclid / Initial permutation
2-OPT
Random
PRD [%]
n
0
50
100
150
200
250
300
350
400
450

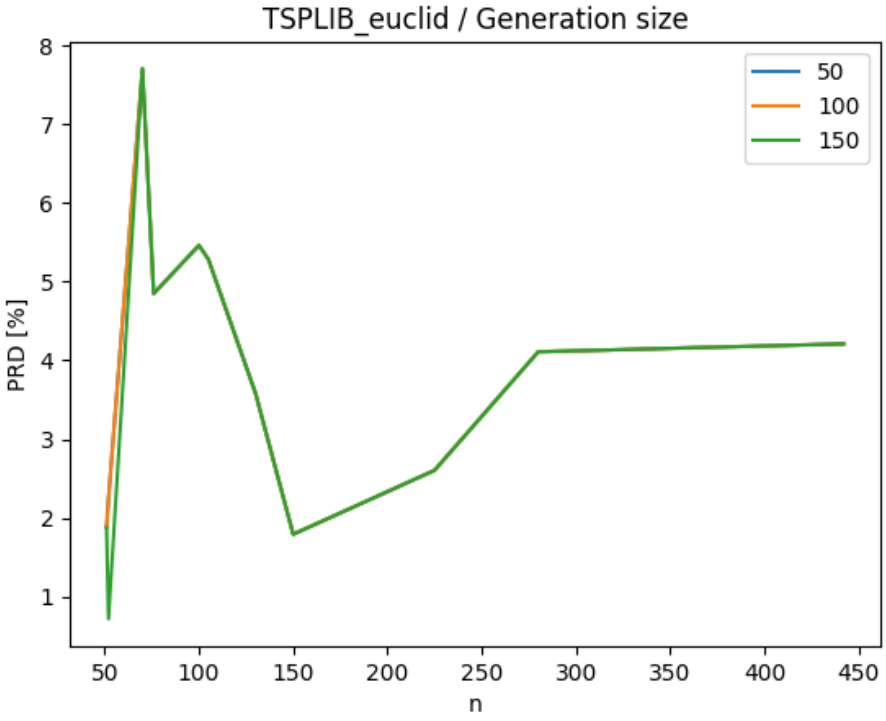

Asym / Initial permutation
2-OPT
Random
PRD [%]
n
0
100
200
300
400
50
100
150
200
250
300
350
400

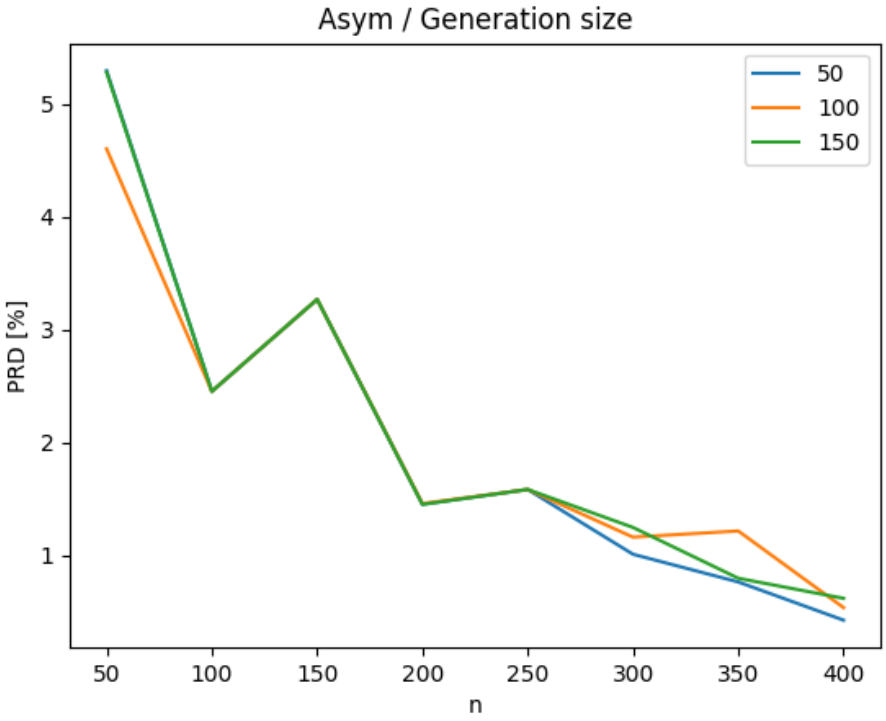

Euclid / Initial permutation
2-OPT
Random
PRD [%]
n
0
25
50
75
100
125
150
175
200
50
100
150
200
250
300
350
400

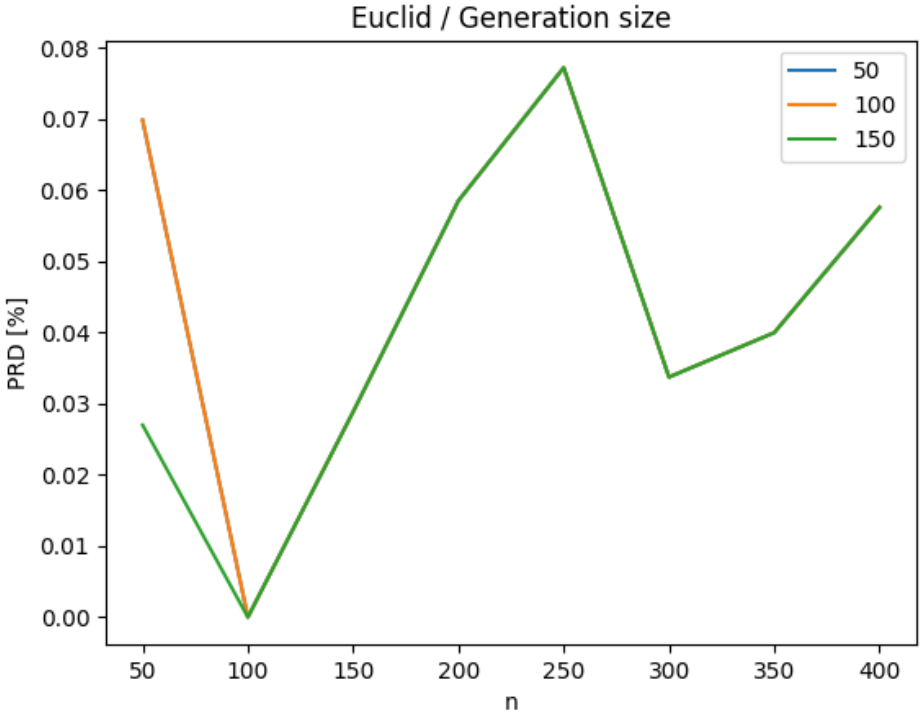

Sym / Initial permutation
2-OPT
Random
PRD [%]
n
0
100
200
300
400
500
600
50
100
150
200
250
300
350
400

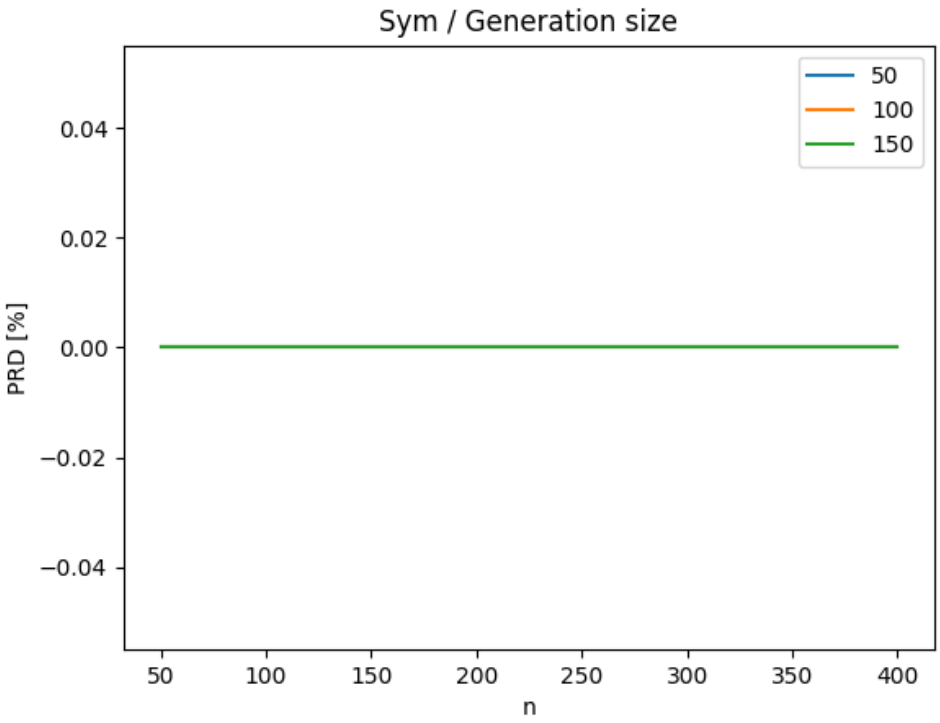
4.3 Wielkość pokolenia



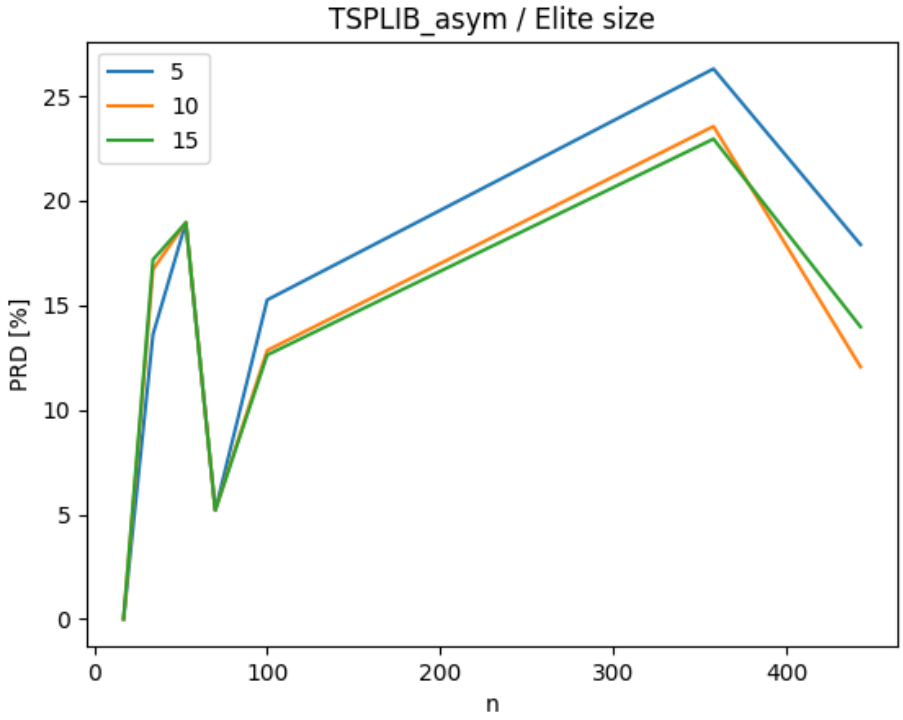


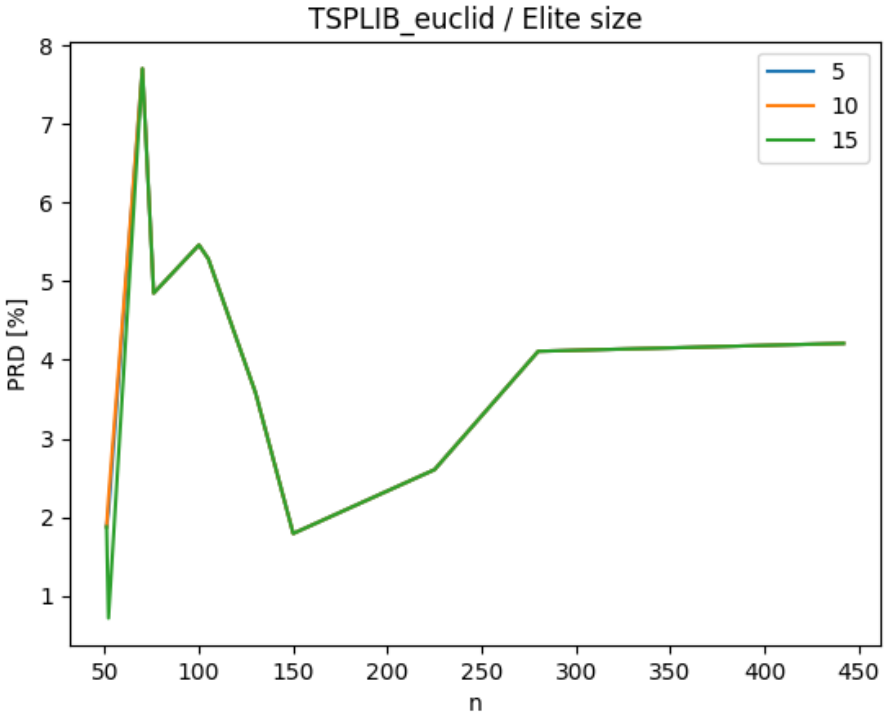

Asym / Generation size
50
100
150
PRD [%]
n
1
2
3
4
5
50
100
150
200
250
300
350
400

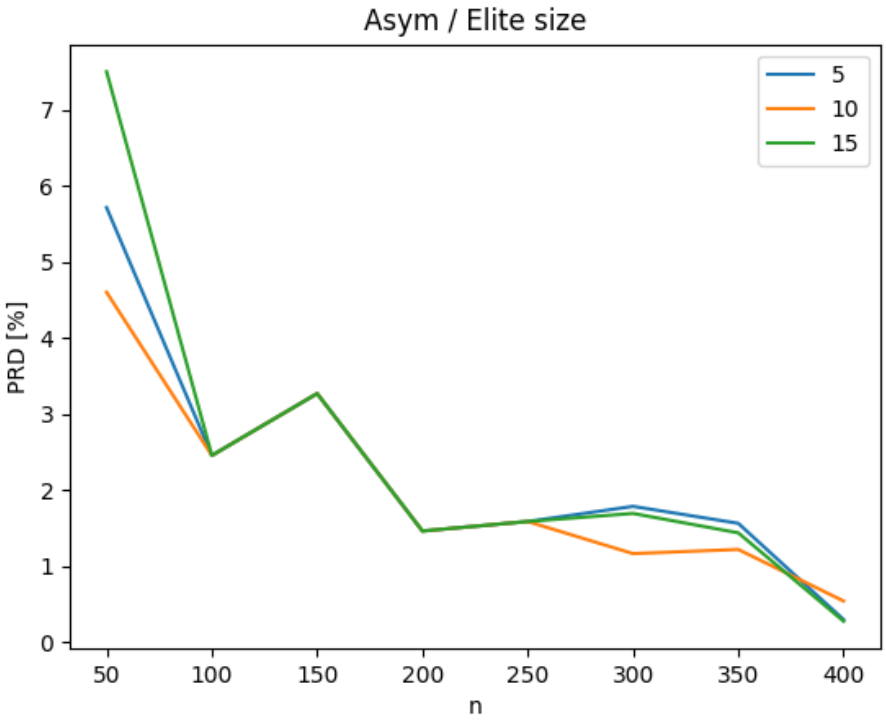

Euclid / Generation size
50
100
150
PRD [%]
n
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
50
100
150
200
250
300
350
400

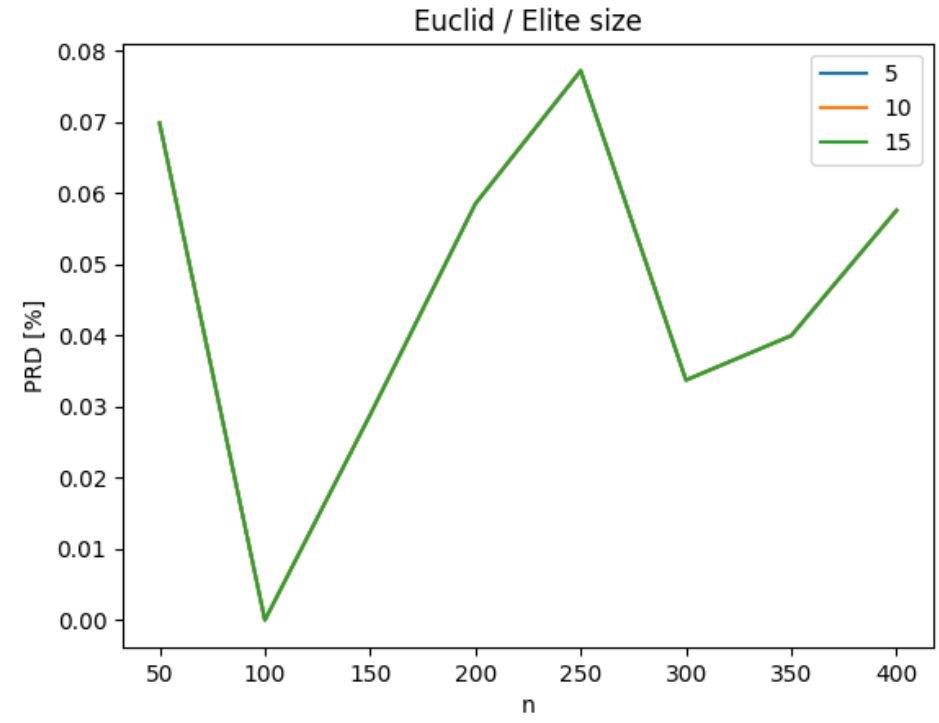


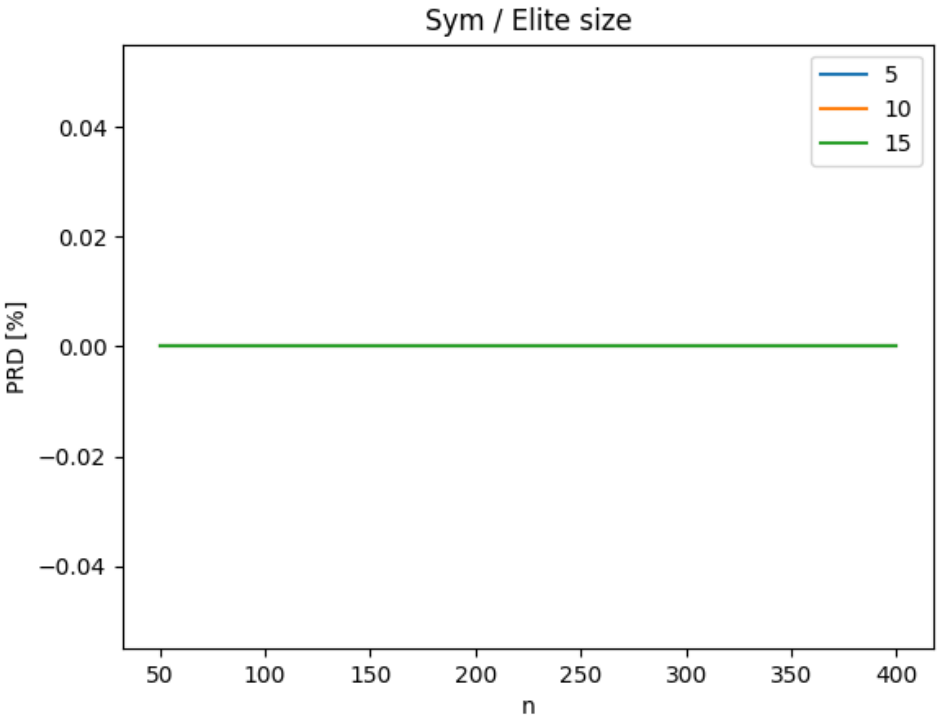
4.4 Wielkość elity



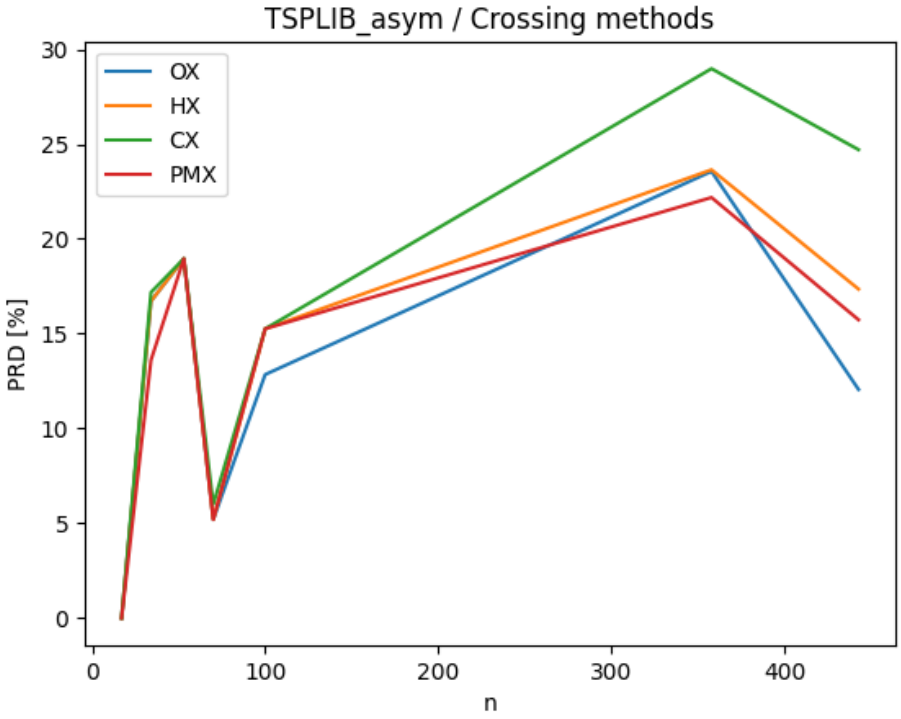

TSPLIB_euclid / Elite size
PRD [%]
n
5
10
15
1
2
3
4
5
6
7
8
50
100
150
200
250
300
350
400
450

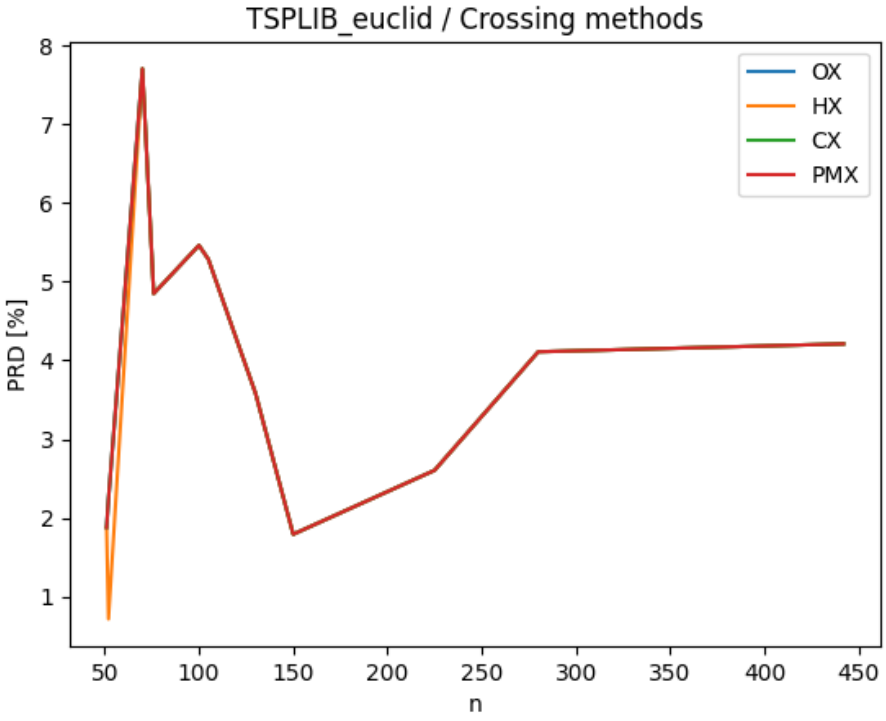

Asym / Elite size
PRD [%]
n
5
10
15
0
1
2
3
4
5
6
7
50
100
150
200
250
300
350
400

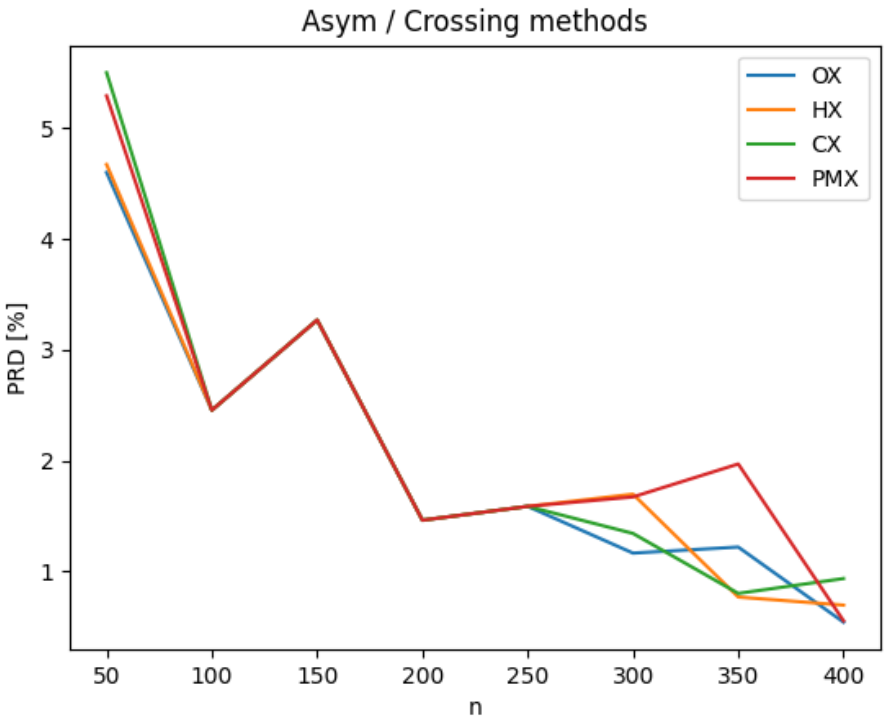

Euclid / Elite size
PRD [%]
n
5
10
15
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
50
100
150
200
250
300
350
400

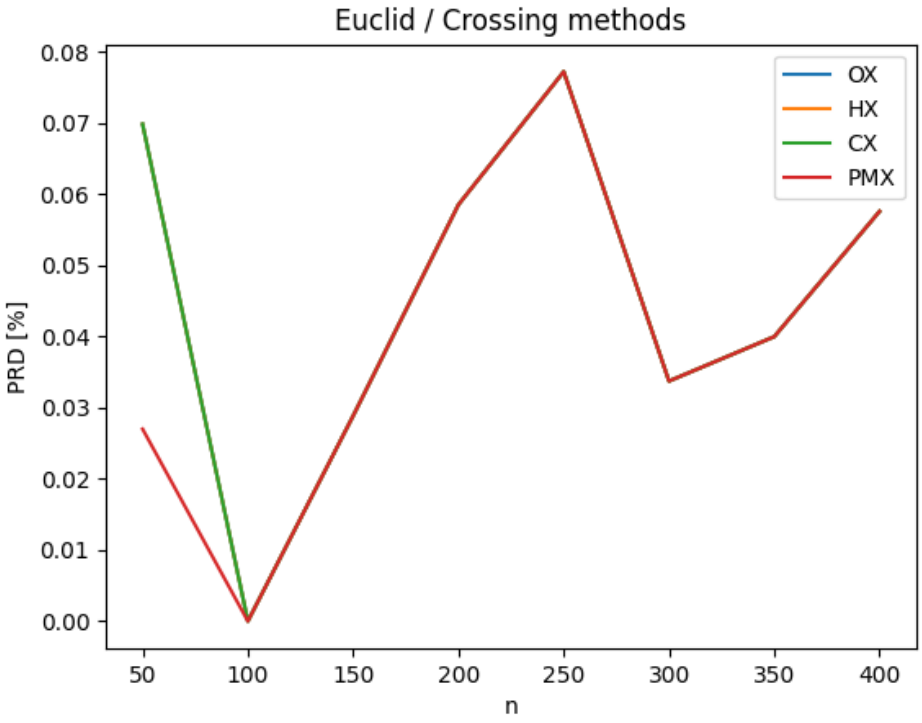

Sym / Elite size
PRD [%]
n
5
10
15
0.04
0.02
0.00
−0.02
−0.04
50
100
150
200
250
300
350
400

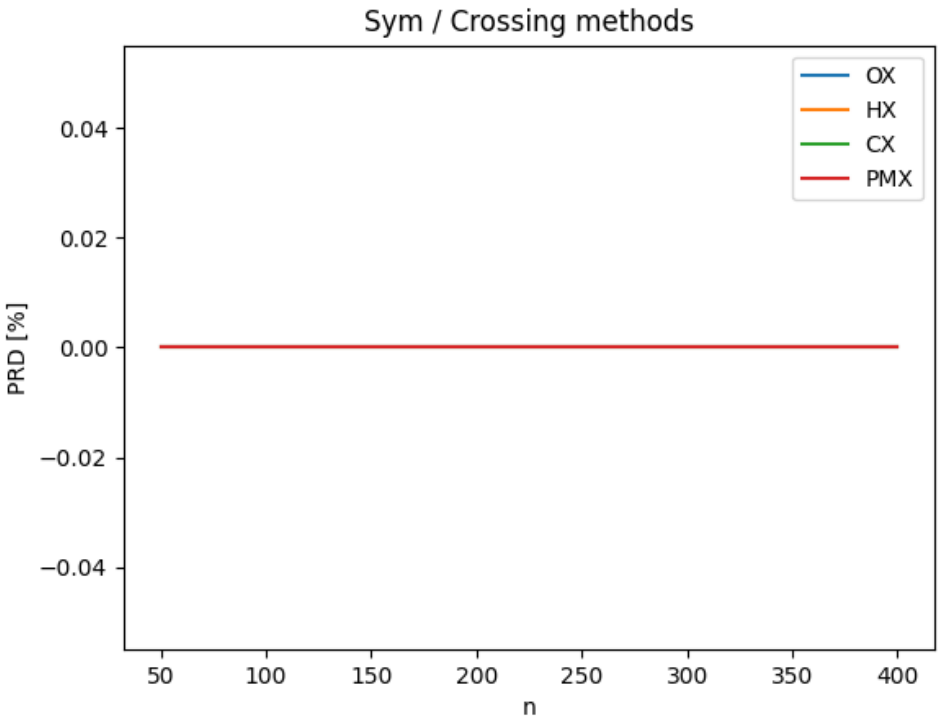
4.5 Krzyżowanie



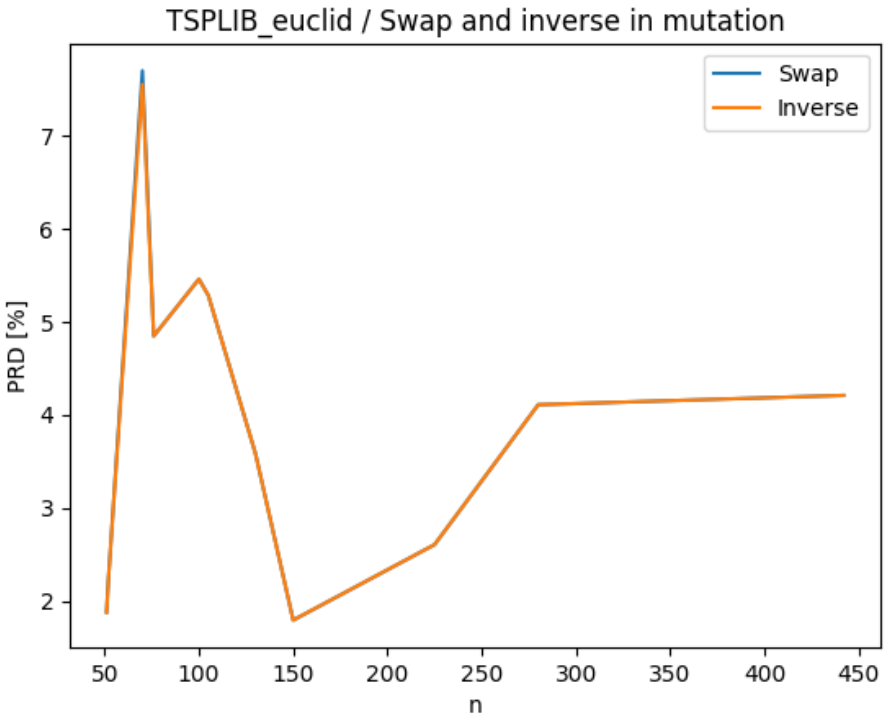


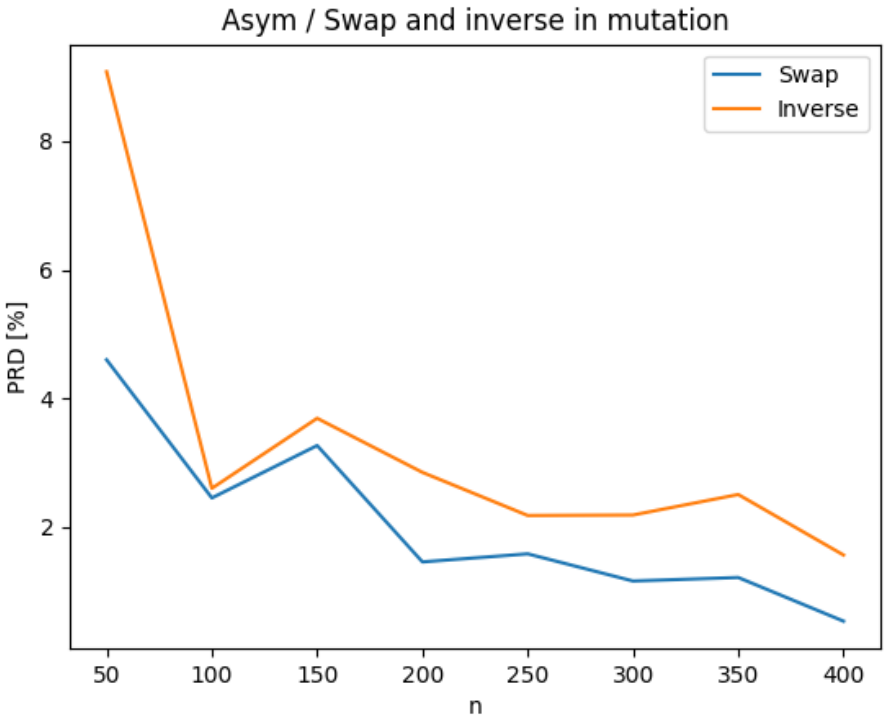

Asym / Crossing methods
OX
HX
CX
PMX
PRD [%]
n
1
2
3
4
5
50
100
150
200
250
300
350
400


Euclid / Crossing methods
OX
HX
CX
PMX
PRD [%]
n
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
50
100
150
200
250
300
350
400

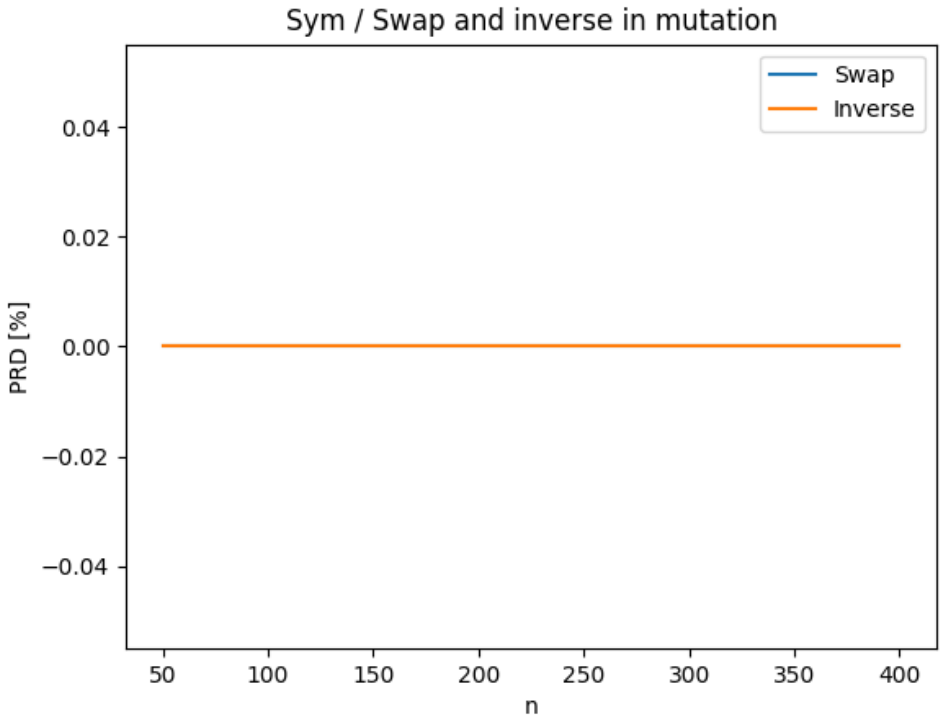


4.6 Swap i inverse

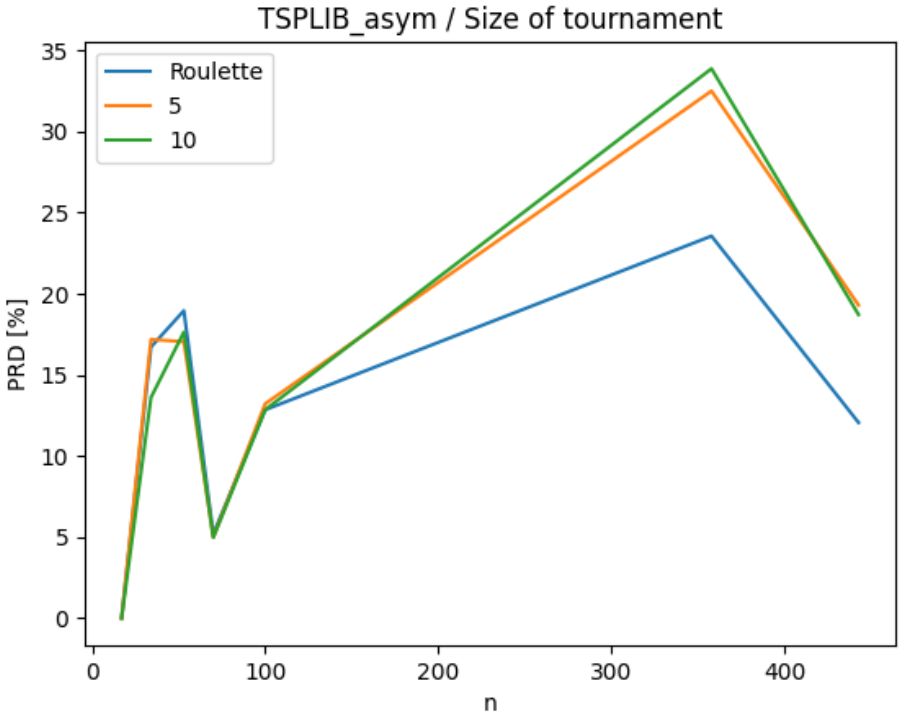

TSPLIB_euclid / Swap and inverse in mutation
Swap
Inverse
PRD [%]
n
2
3
4
5
6
7
50
100
150
200
250
300
350
400
450

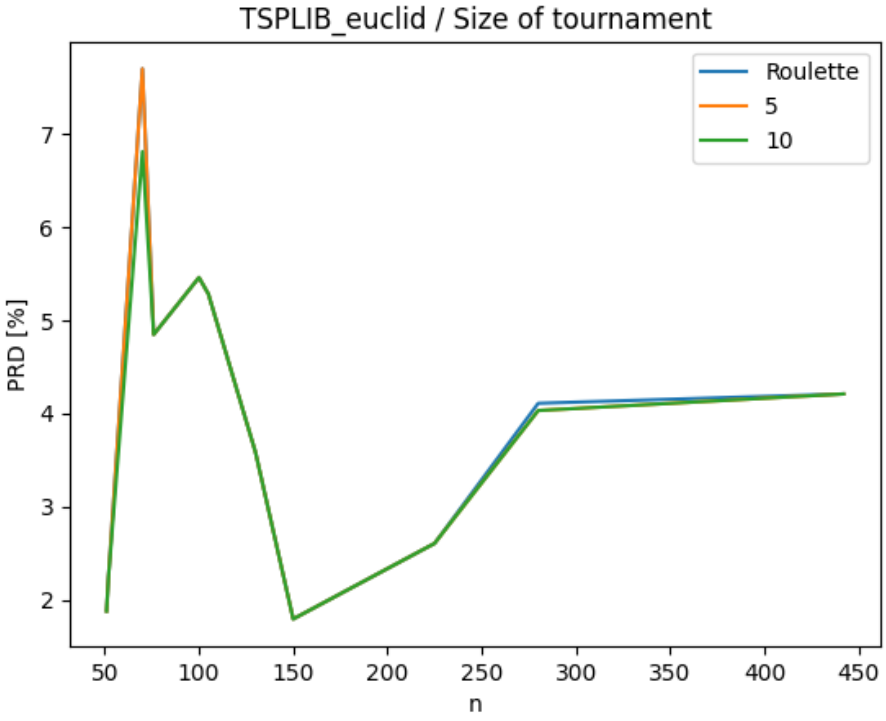

Asym / Swap and inverse in mutation
Swap
Inverse
PRD [%]
n
2
4
6
8
50
100
150
200
250
300
350
400

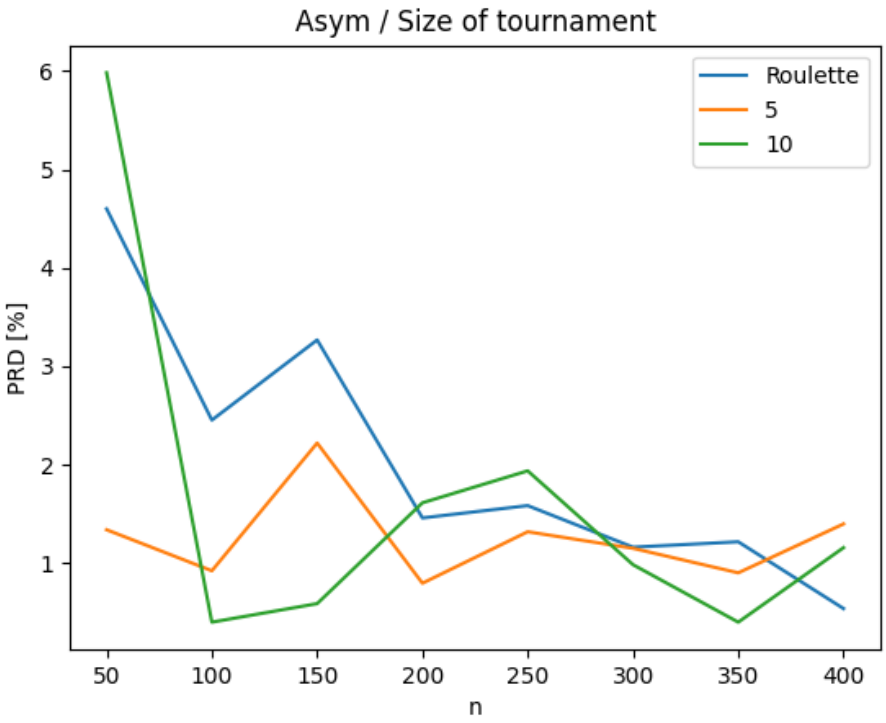

Euclid / Swap and inverse in mutation
Swap
Inverse
PRD [%]
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
50
100
150
200
250
300
350
400
n

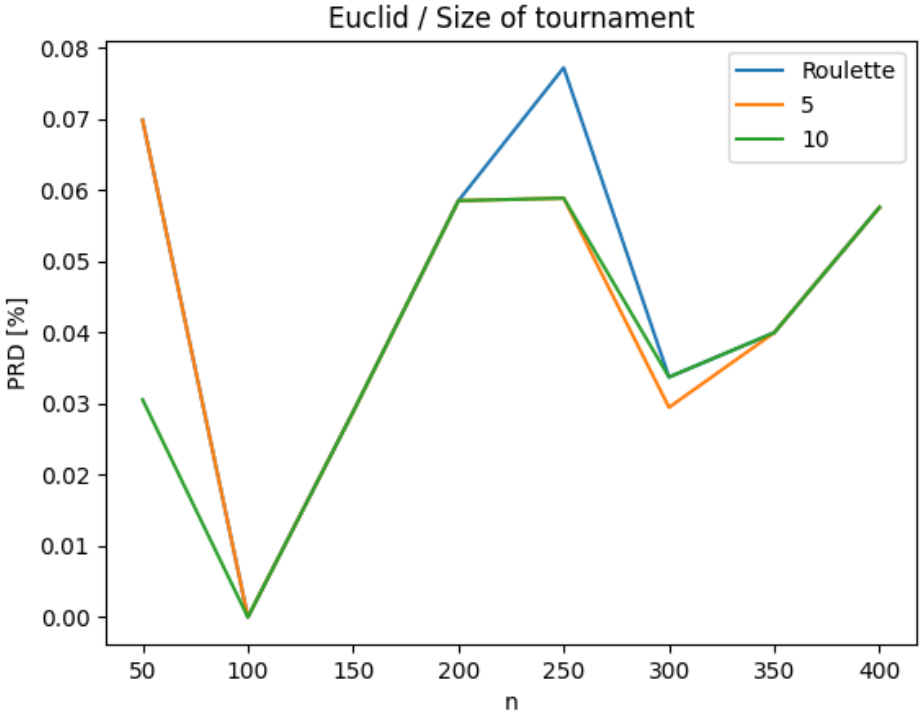

Sym / Swap and inverse in mutation
Swap
Inverse
PRD [%]
0.04
0.02
0.00
−0.02
−0.04
50
100
150
200
250
300
350
400
n

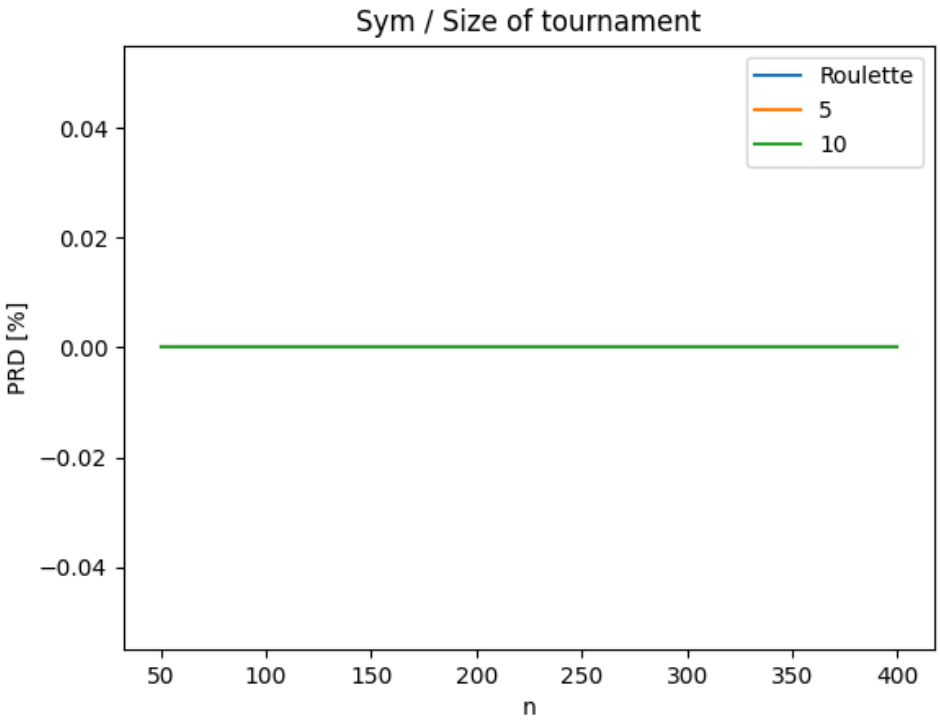
4.7 Turniej


TSPLIB_asym / Size of tournament
Roulette
5
10
PRD [%]
n
0
5
10
15
20
25
30
35
0
100
200
300
400

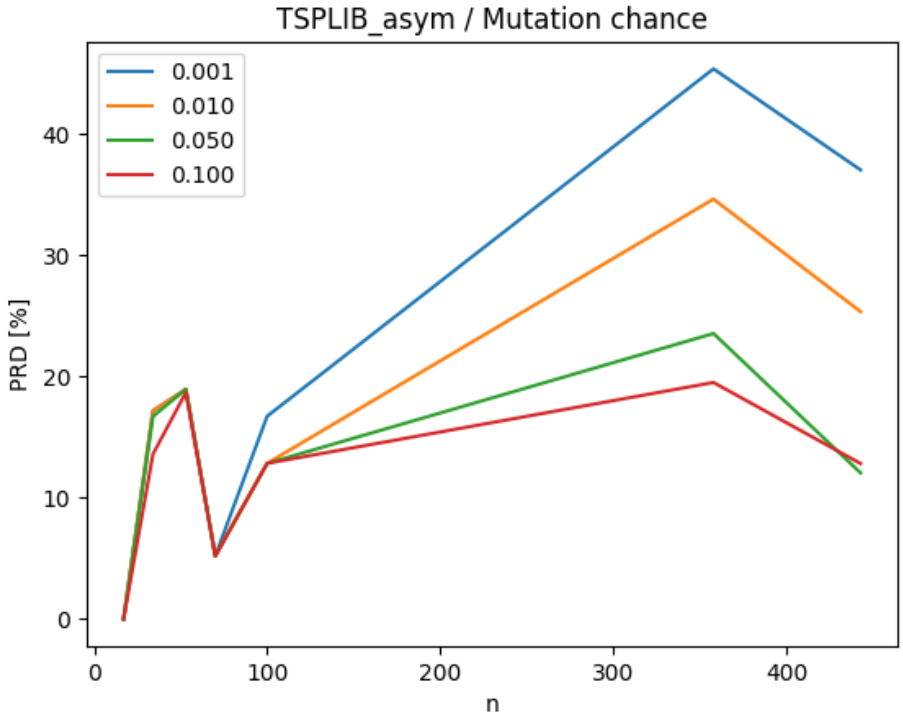

TSPLIB_euclid / Size of tournament
Roulette
5
10
PRD [%]
n
2
3
4
5
6
7
50
100
150
200
250
300
350
400
450

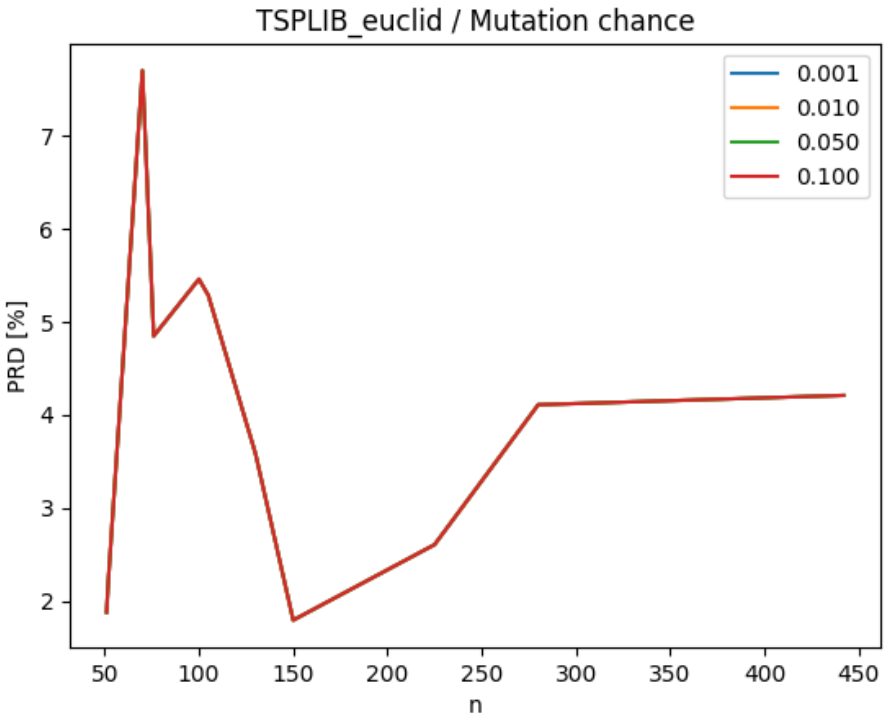

Asym / Size of tournament
Roulette
5
10
PRD [%]
n
6
5
4
3
2
1
50
100
150
200
250
300
350
400

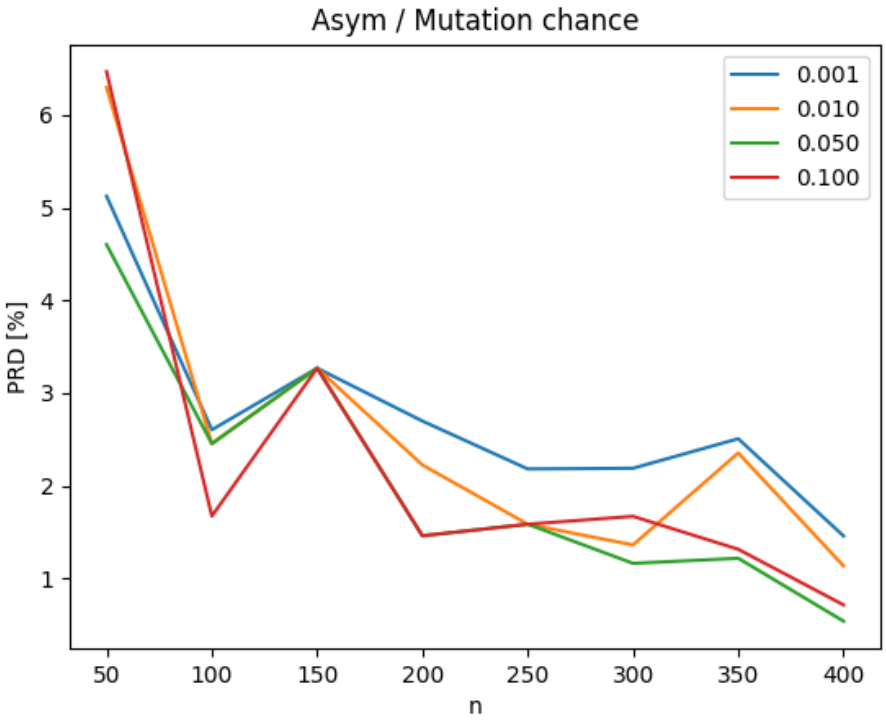

Euclid / Size of tournament
Roulette
5
10
PRD [%]
n
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
50
100
150
200
250
300
350
400

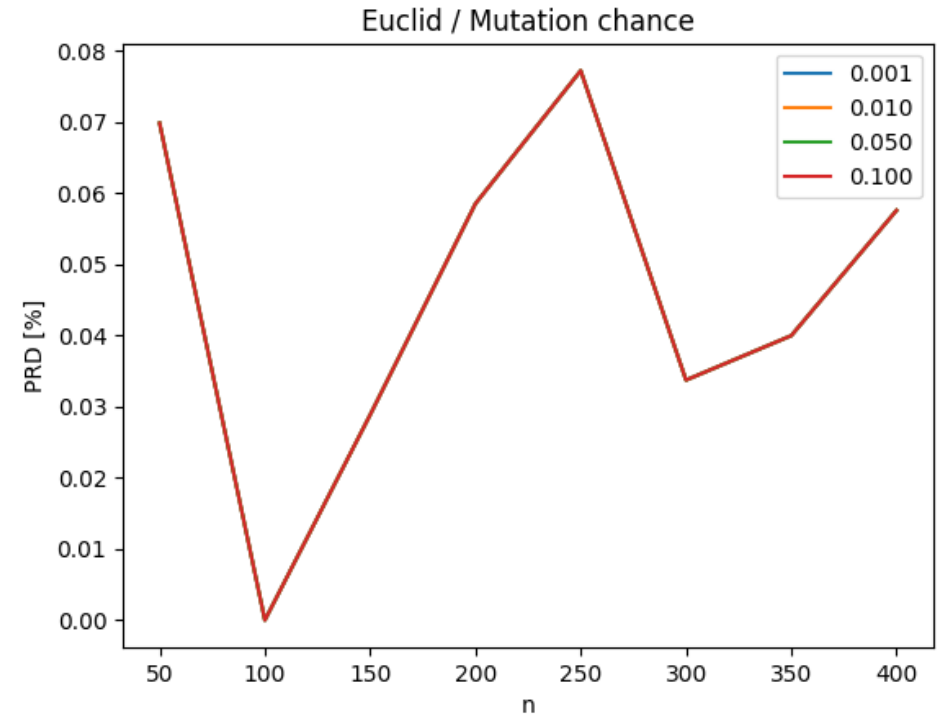


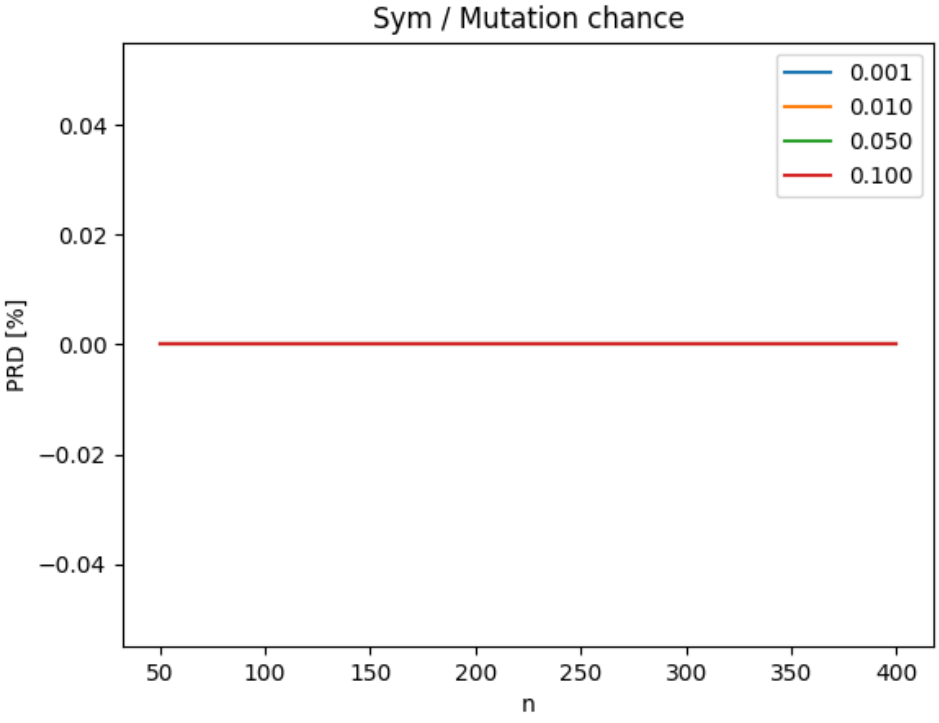
4.8 Szansa mutacji



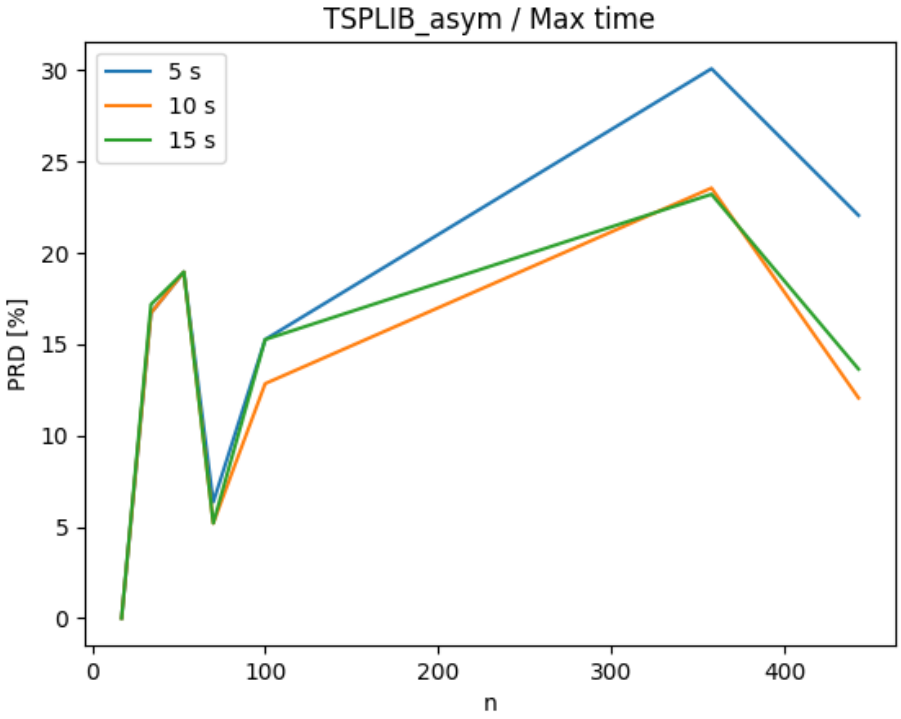

TSPLIB_euclid / Mutation chance
0.001
0.010
0.050
0.100
PRD [%]
n
2
3
4
5
6
7
50
100
150
200
250
300
350
400
450

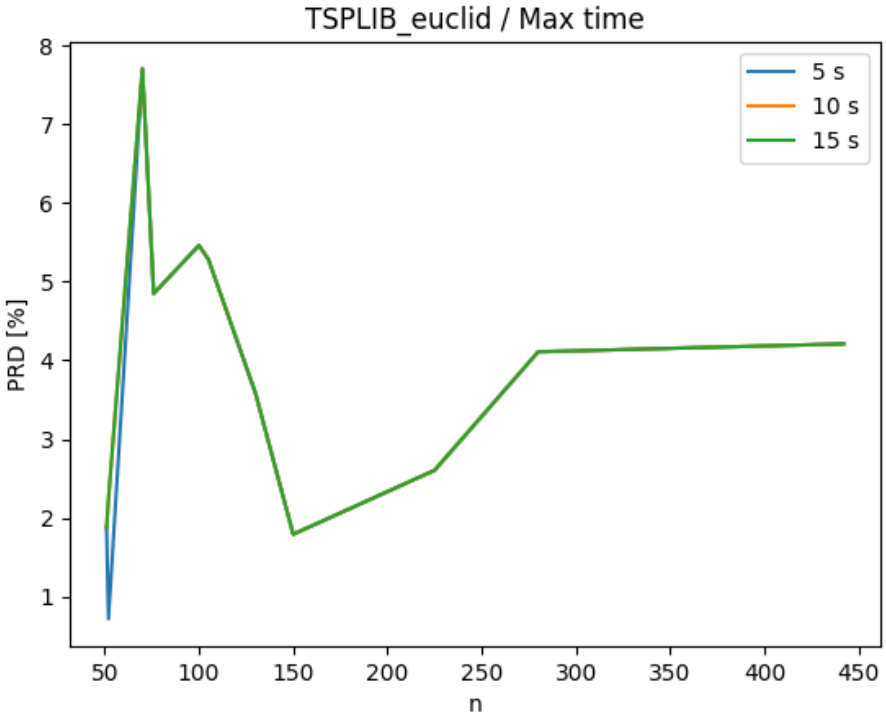

Asym / Mutation chance
0.001
0.010
0.050
0.100
PRD [%]
n
1
2
3
4
5
6
50
100
150
200
250
300
350
400

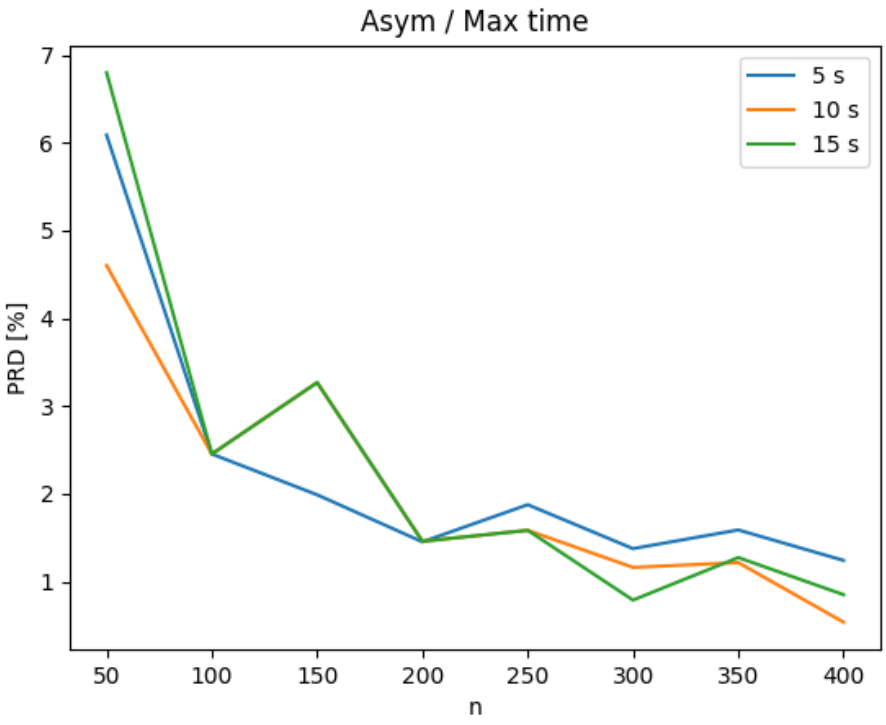

Euclid / Mutation chance
0.001
0.010
0.050
0.100
PRD [%]
n
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
50
100
150
200
250
300
350
400

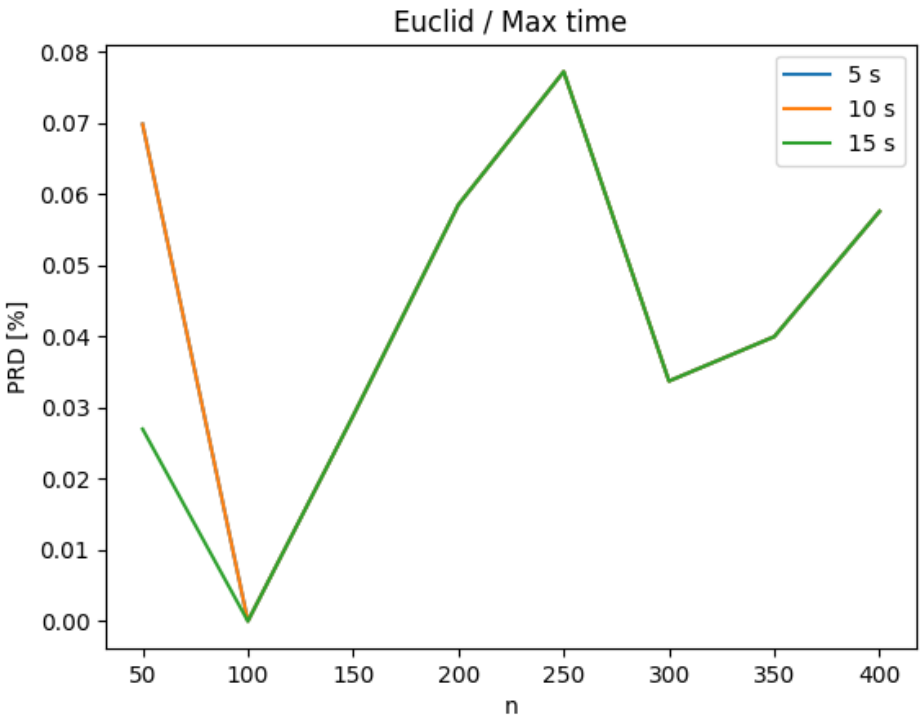

Sym / Mutation chance
0.001
0.010
0.050
0.100
PRD [%]
n
−0.04
−0.02
0.00
0.02
0.04
50
100
150
200
250
300
350
400

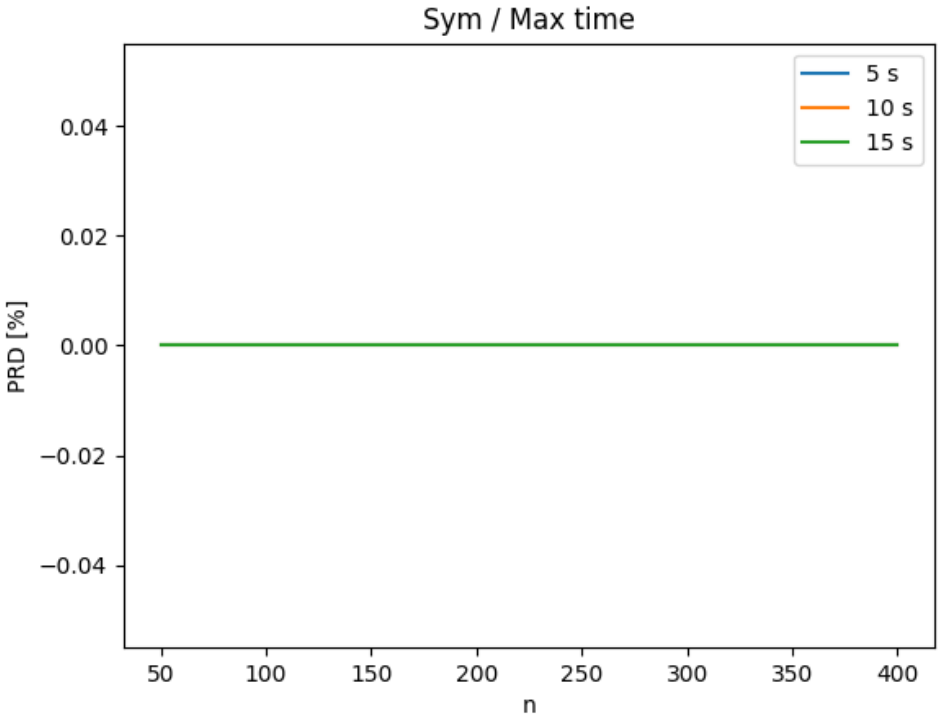
4.9 Maksymalny czas



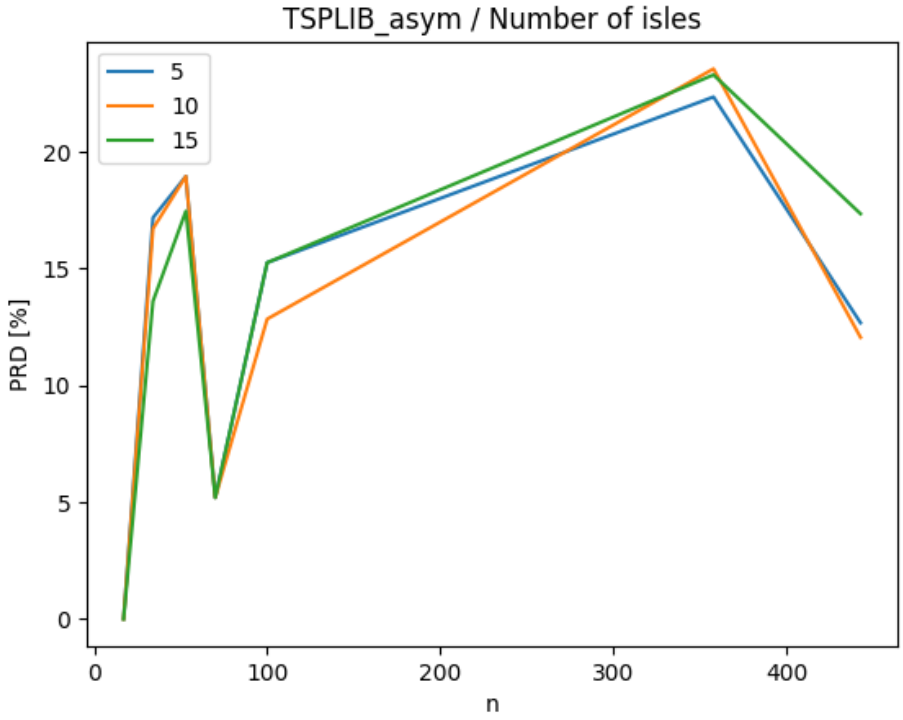


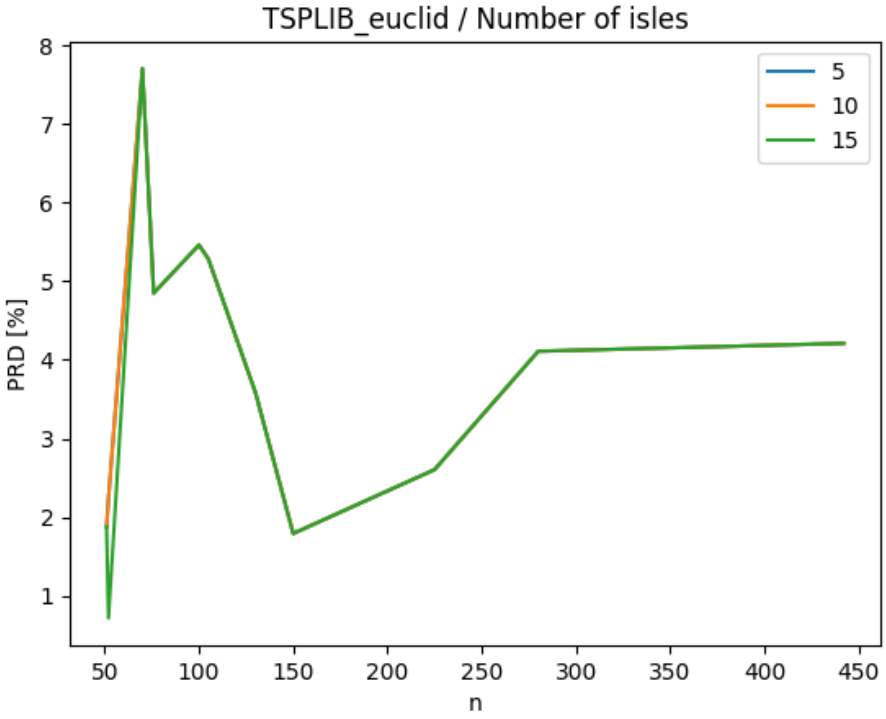

Asym / Max time
PRD [%]
n
5 s
10 s
15 s
7
6
5
4
3
2
1
50
100
150
200
250
300
350
400

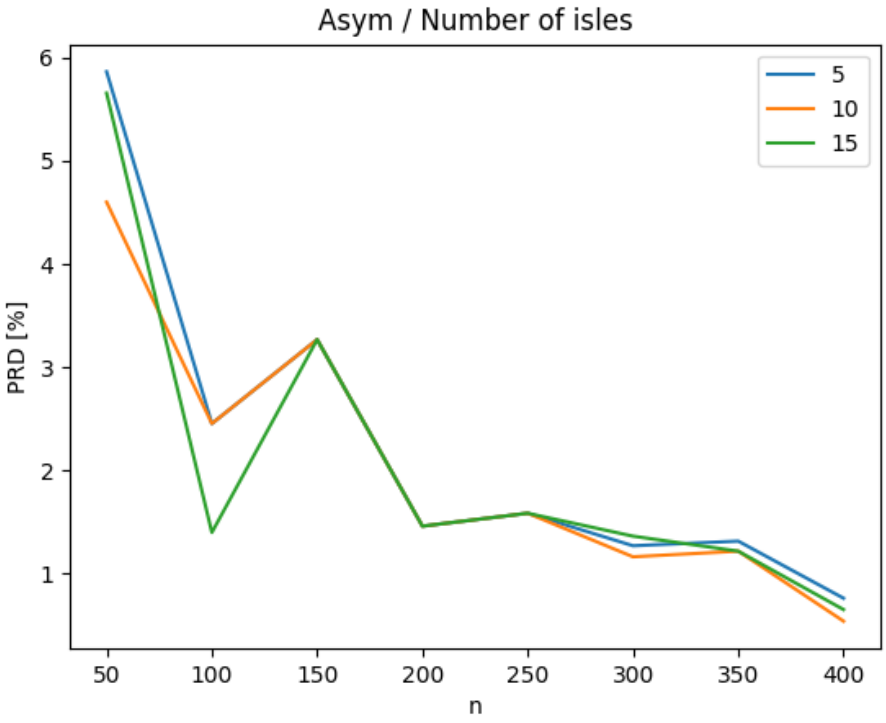

Euclid / Max time
PRD [%]
n
5 s
10 s
15 s
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
50
100
150
200
250
300
350
400

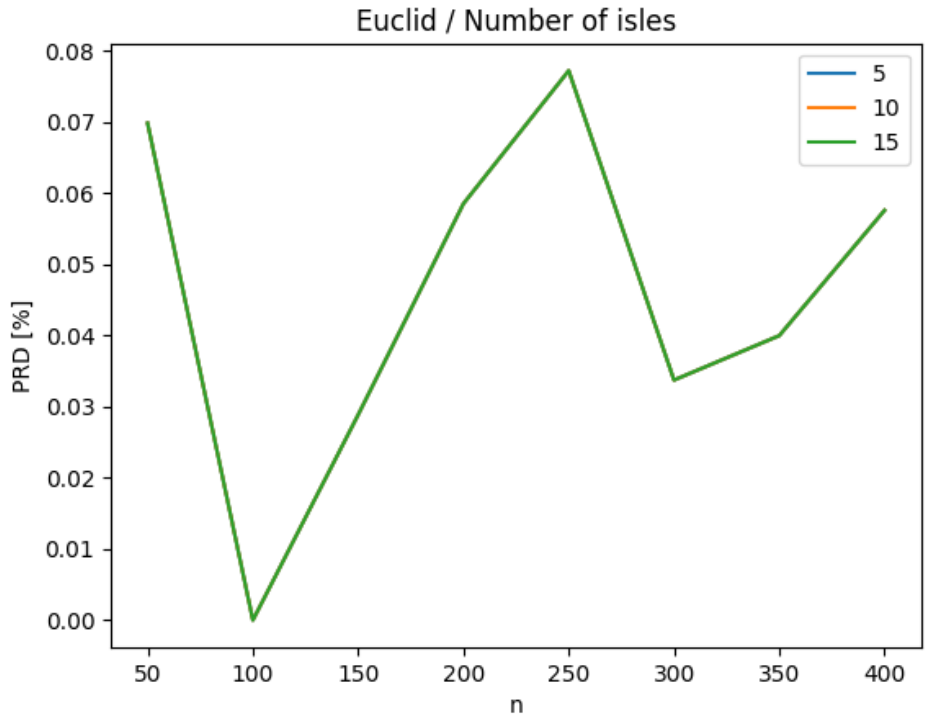


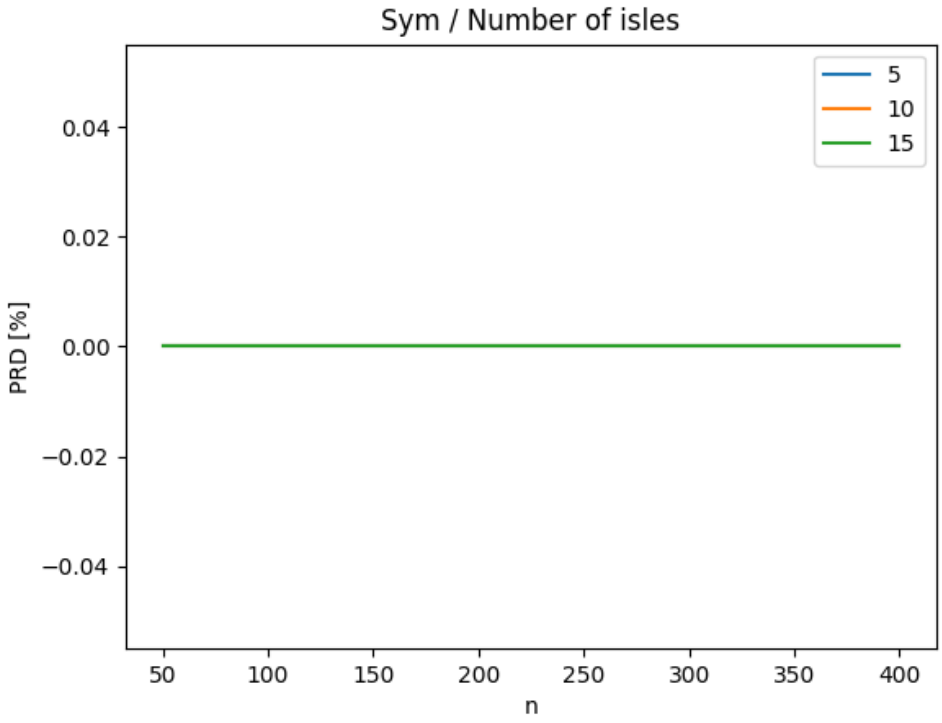
4.10 Liczba wysp



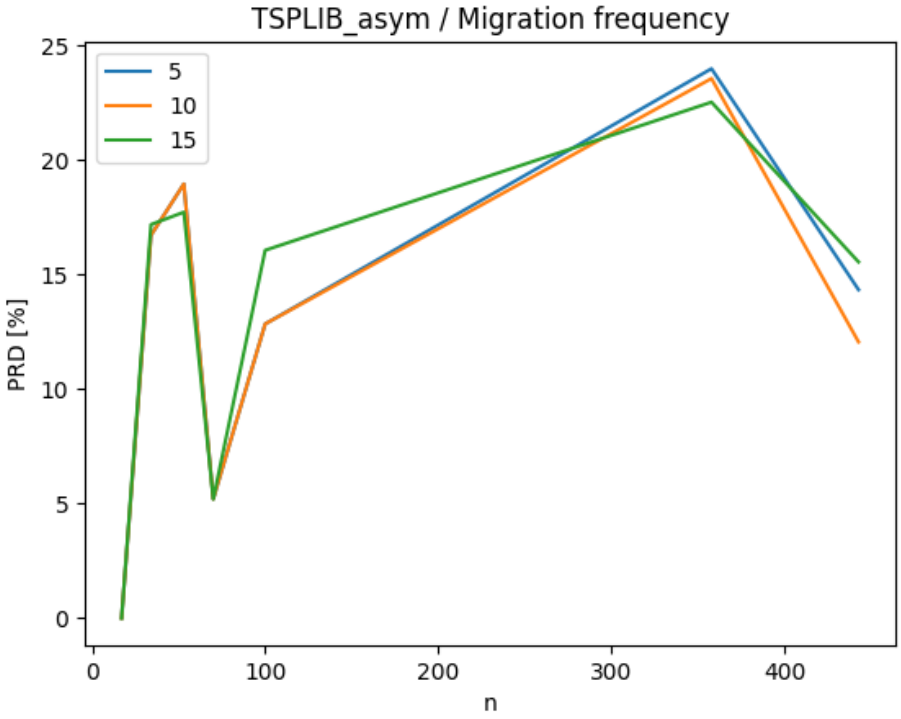

TSPLIB_euclid / Number of isles
PRD [%]
n
5
10
15
1
2
3
4
5
6
7
8
50
100
150
200
250
300
350
400
450

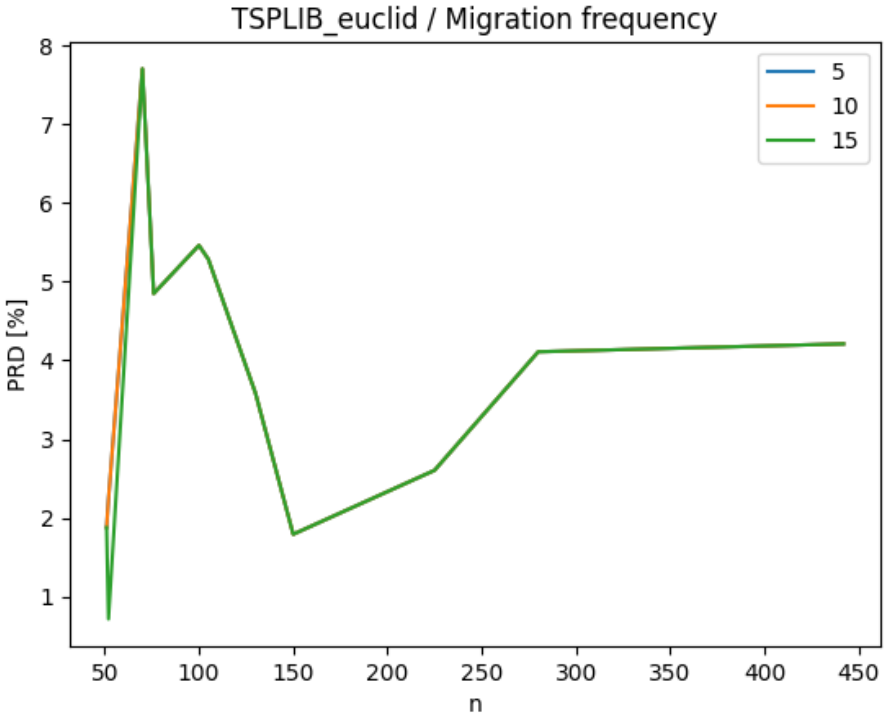

Asym / Number of isles
PRD [%]
n
5
10
15
1
2
3
4
5
6
50
100
150
200
250
300
350
400

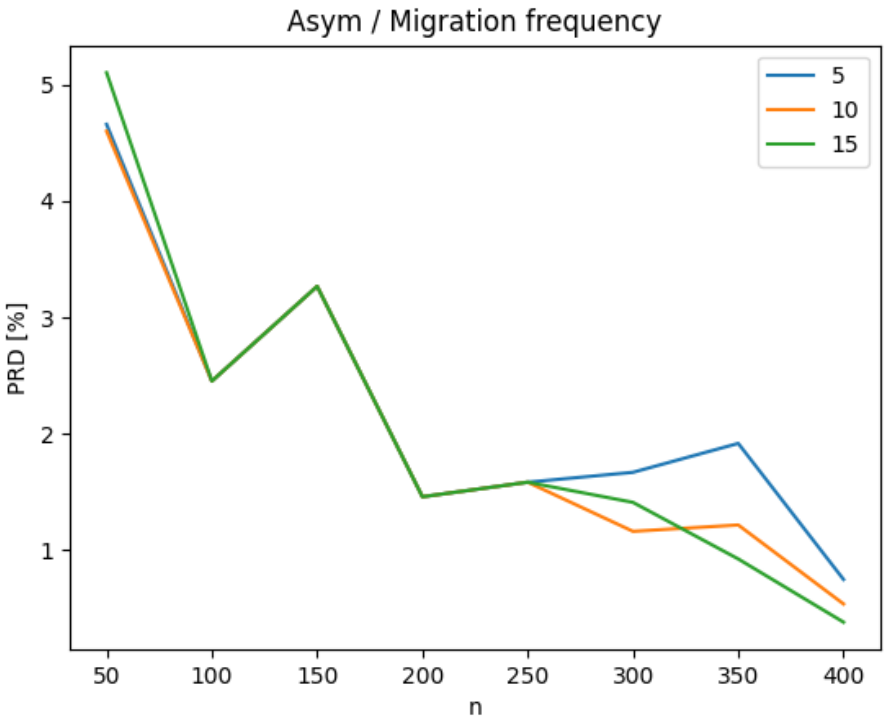

Euclid / Number of isles
PRD [%]
n
5
10
15
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
50
100
150
200
250
300
350
400

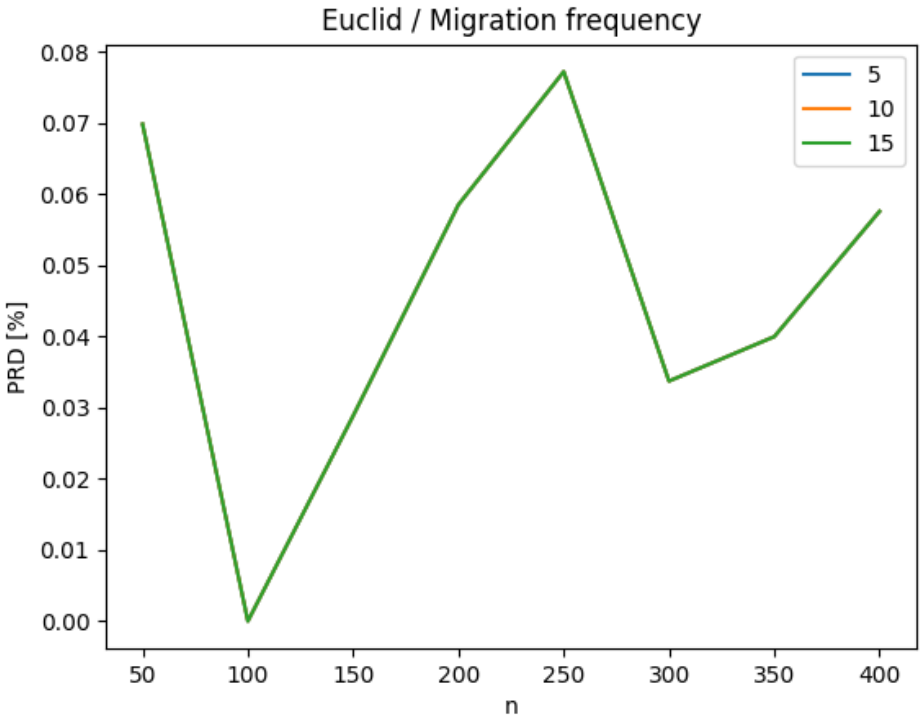

Sym / Number of isles
PRD [%]
n
5
10
15
0.04
0.02
0.00
−0.02
−0.04
50
100
150
200
250
300
350
400

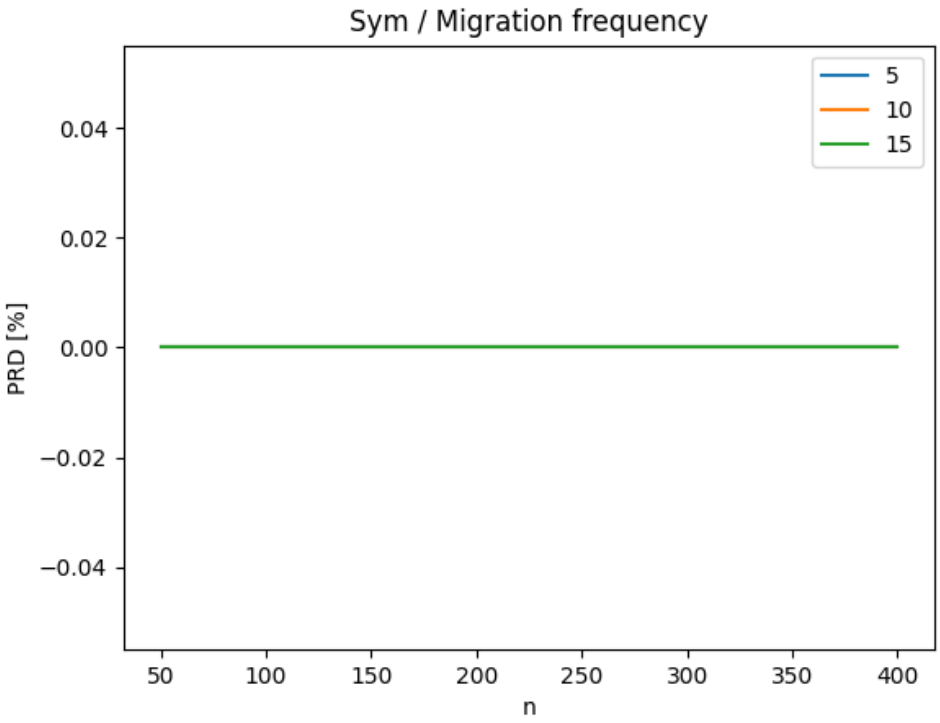
4.11 Częstotliwość migracji



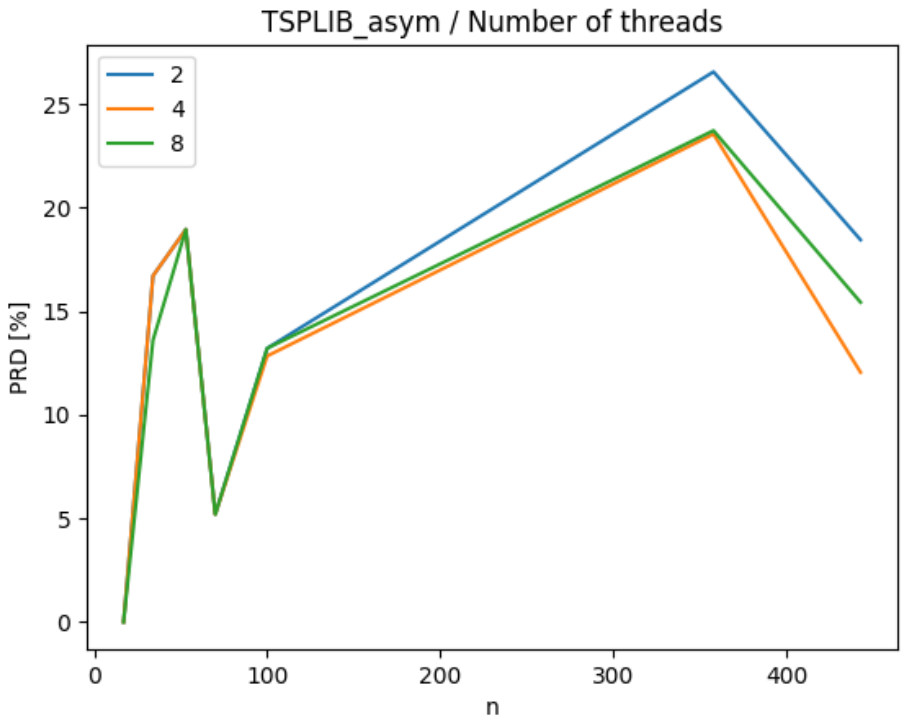


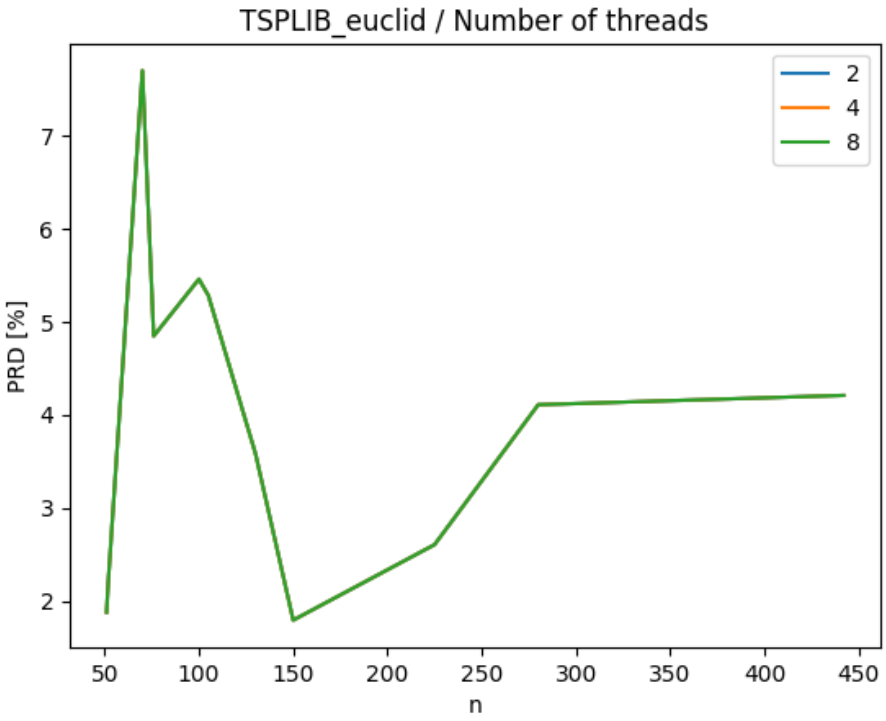

Asym / Migration frequency
PRD [%]
n
5
10
15
1
2
3
4
5
50
100
150
200
250
300
350
400

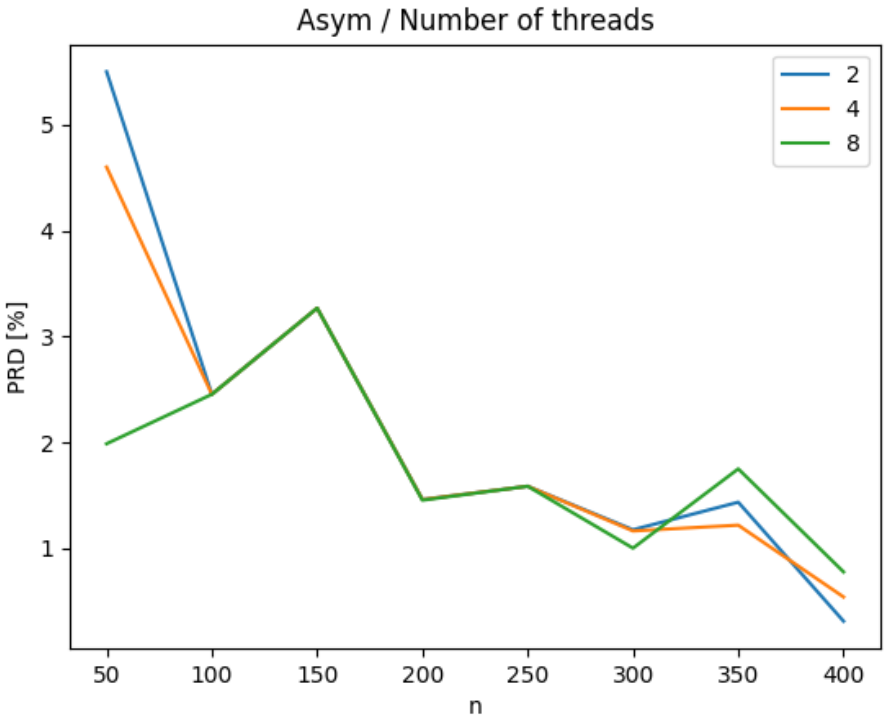

Euclid / Migration frequency
PRD [%]
n
5
10
15
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
50
100
150
200
250
300
350
400

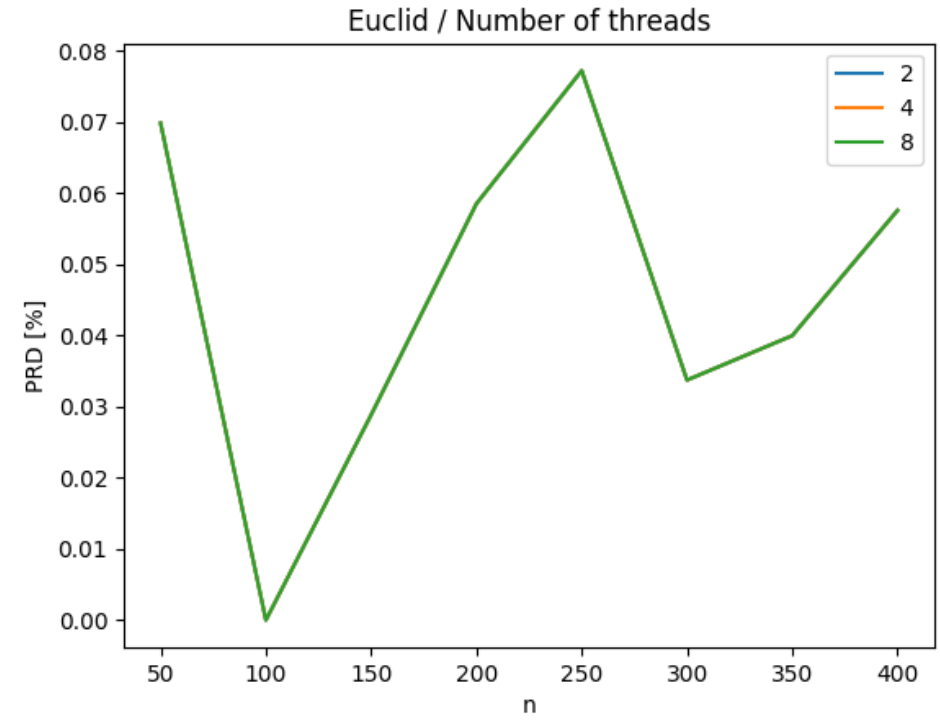


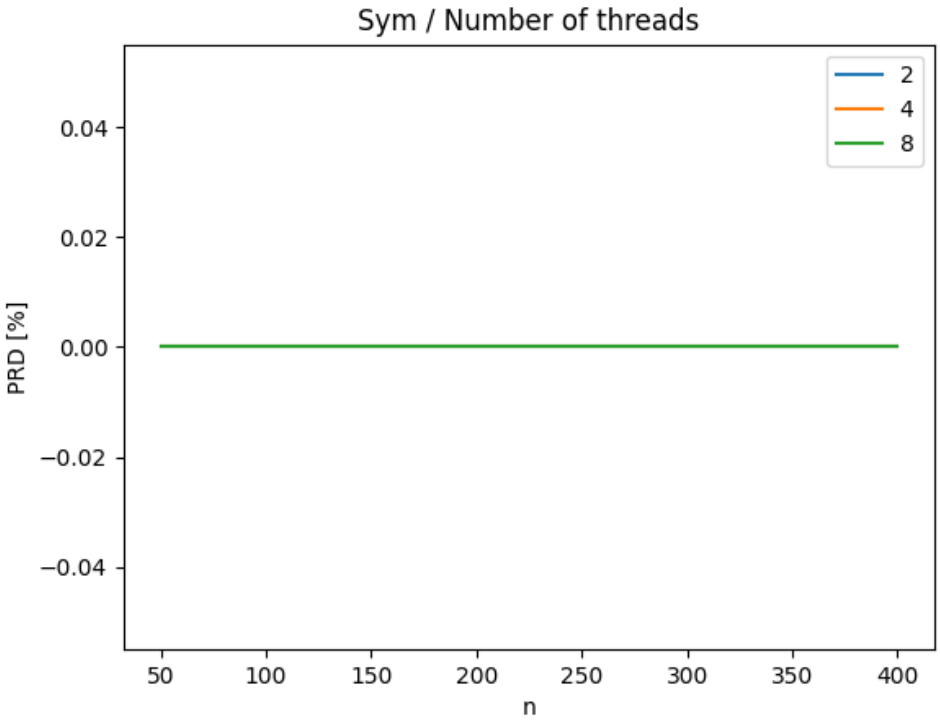
4.12 Liczba wątków




TSPLIB_euclid / Number of threads
2
4
8
PRD [%]
n
2
3
4
5
6
7
50
100
150
200
250
300
350
400
450


Asym / Number of threads
2
4
8
PRD [%]
n
1
2
3
4
5
50
100
150
200
250
300
350
400





5 Obserwacje i wnioski

- W przypadku praktycznie wszystkich testowanych macierzy, algorytm genetyczny działał tak samo dobrze lub gorzej od zaimplementowanego na liście 2 przeszukiwania tabu. Pozwala to stwierdzić, że dla problemu komiwojażera najbardziej opłaca się użyć tabu search (jeśli chcemy otrzymać najlepsze wyniki) lub 2-opta (jeśli zależy nam na czasie, ale wyniki wciąż mają być stosunkowo dobre).
- Dla każdego typu macierzy powtórzył się jeden wynik - jeśli pierwsze pokolenie było całkowicie losowe, to dawało to bardzo słabe rezultaty, nawet nie zbliżające się do optymalnych. Natomiast przy przybliżeniu pierwszego pokolenia innymi

metaheurystykami(2-OPT) wyniki stawały się bardzo dobre.

- W przypadku macierzy symetrycznych, podobnie jak w eksperymentach z poprzedniej listy, nie ma różnicy w wynikach w zależności od wybranych parametrów. Dla tych macierzy optymalnym rozwiązaniem jest 2-opt, ponieważ algorytmy startujące z przybliżenia początkowego generowanego przez 2-opt nie poprawiają wyników.

- Dla macierzy euklidesowych różnice między poszczególnymi algorytmami i wybranymi parametrami istnieją, ale są minimalne.

- Macierze asymetryczne dostarczają znacznie bardziej różnorodnych wyników:

  Najlepszy okazał się algorytm z wątkami i wyspami, chociaż dla mniejszych macierzy generowanych przez nas najlepszy okazał się algorytm bez wątków oraz wysp, pokonując nawet tabu search.

  Najlepszy rozmiar macierzy dla danych z TSPLIB to 100, osiągając podobny wynik do 50, natomiast największy testowany rozmiar - 150 - miał znacznie gorsze wyniki. Dla danych generowanych wyniki były bardziej do siebie zbliżone i ciężko jednoznacznie określić zwycięzcę.

  Algorytm dawał najlepsze wyniki dla większych wielkości elity (10 i 15 na 100-osobowe pokolenie), niż mniejszej (5).

  Średnio najlepszą metodą krzyżowania okazała się Order Crossover.

  Metoda swap stosowana przy mutacji dawała znacznie lepsze rezultaty niż inverse (często nawet dwukrotnie lepsze).

  W danych z TSPLIB zdecydowanie najlepiej wypadł turniej z wykorzystaniem zasady ruletki, niż klasyczny turniej z ustalonymi wielkościami. Co ciekawe, na danych generowanych przez nas wyniki ruletki były dla wielu n najgorsze.

  Zwiększanie szansy mutacji do pewnego stopnia poprawiało otrzymywane wyniki - najwyższa szansa mutacji (0.1) dawała najlepsze wyniki.

  Ograniczenie czasowe nie wpływało aż tak na wyniki - o ile najkrótszy czas (5 sekund) dawał najgorsze rezultaty, to pomiędzy 10 i 15 sekund nie było aż takiej różnicy, a wyniki były nawet często lepsze dla średniego czasu (10s).

  Liczba wysp dawała zawsze zbliżone wyniki dla 5, 10 i 15 wysp, często różnie dla róznych wielkości macierzy, zatem ciężko wybrać najlepszą wartość.

  Optymalną częstotliwością migracji na podstawie wykresów można nazwać 10.

  Optymalną liczbą wątków są 4, ponieważ w przypadku większej liczby wątków obliczenia i tak nie wykonają się równolegle, a komunikacja między wątkami kosztuje.