Laboratorium Badań Operacyjnych

Sprawozdanie końcowe

Problem przydziału pracowników do stanowisk na określony przedział czasu

Jakub Kołton

Antoni Kraczowski

06 I 2024

Spis treści

Podział pracy

Etap	Jakub Kołton	Antoni Kraczowski
Model zagadnienia	5%	95%
Algorytm opracowanie	95%	5%
Implementacja aplikacji	80%	20%
Testy	100%	0%
Dokumentacja	10%	90%

Model zagadnienia

Problem polega na optymalnym przydziale pracowników do maszyn w fabryce. Fabryka zawiera *M* maszyn i *P* pracowników. Każda maszyna ma określony koszt pracy na jednostkę czasu, maksymalną ilość pracowników potrzebnych do obsługi oraz ilość produkowanego towaru w jednostce czasu. Dodatkowo każda maszyna produkuje inny rodzaj towaru. Każdy pracownik ma ustaloną stawkę godzinową, określone doświadczenie na każdej z maszyn i ilość godzin jaką pracuje na jednej zmianie (4/8 godzin). Fabryka dostaje zamówienie na dane ilości każdego z towarów w zadanym terminie wyrażonym w godzinach. Zadaniem algorytmu jest przydzielenie pracowników na takie zmiany i do takich stanowisk aby wykonać zlecenie jak najmniejszym kosztem jednocześnie mieszcząc się w terminie.

Rozwiązaniem jest macierz zer i jedynek X = M x P x T

M – liczba maszyn; P – liczba pracowników; T – liczba jednostek czasów rozkładzie;

1 w macierzy oznacza że na maszynie o indeksie m pracuje pracownik p w godzinie t.

Koszt rozwiązania wyrażany jest w następujący sposób:

$$K = m(X) + s(X) + t(X)$$

gdzie kolejno:

m – koszt pracy maszyn

s – koszt utrzymania pracowników

t – kara za niewywiązanie się z ustalonego terminu

$$m(X) = A^T \times \sum_{k=0}^{T} sgn(\sum_{j=0}^{P} X(:, j, k))$$

A – macierz n x 1 kosztów pracy maszyn; sgn() to funkcja signum

W ten sposób otrzymuje się koszt pracy wszystkich maszyn zależny od tego ile czasu pracował przy niej przynajmniej jeden pracownik.

$$s(X) = \sum_{k=0}^{T} (\sum_{i=0}^{M} (X(i, :, k))) \times B$$

B – macierz n x 1 kosztów stawek godzinowych pracowników

$$t(X) = a * t_1 + b * t_2$$

gdzie:

a, b – współczynniki kary za ilość dni pracujących i dni przekraczających termin, b >> a;

t_1 – ilość dni pracujących;

t_2 – ilość dni przekraczających termin.

Algorytm

PRAWDOPODOBIEŃSTWO ZMIANY STANU

"The laws of thermodynamics state that at temperature T, the probability of a change in energy magnitude ΔE is given by6 "

$$p(\Delta E) = \exp\left(-\frac{\Delta E}{k_B T}\right) \quad (3)$$

"where ΔE = Ej − Ei, Ei is the current energy state,
Ej is a new energy state and kB is a physical constant known as Boltzmann constant."

ALGORYTM SYMULOWANEGO WYŻARZANIA

Step 1: Randomly initialize x_i within the solution space
Initialize temperature T and set iteration counter $k = 1$
Step 2: Calculate $f(x_i)$
Step 3: Select $x_i' = N(x_i)$
Step 4: Calculate $f(x_i')$
Step 5: Calculate $\Delta E = f(x_i) - f(x_i')$
Calculate $p(x)$ according to Eq. (3)
If $f(x_i') < f(x_i)$
$x_i = x_i'$
else if $[rand[0, 1) < p(x)]$
$x_i = x_i'$
$f(x_i) = f(x_i')$
Step 6: Decrease temperature according to a cooling function $T = g(T, k)$
Step 7: If termination condition is not met, $k = k + 1$ and Go to Step 3
Step 8: Return solution

- k– iterator kroku
- x_i - aktualne rozwiązanie
- $f(x_i)$ - funkcja celu
- $N(x_i)$ - wybór sąsiada rozwiązania
- p(x) - prawdopodobieństwo zmiany stanu
- g(T, k) - funkcja chłodzenia

1. **Solver:** obiekt przechowujący wszystkie parametry, funkcje oraz wartości początkowe wykorzystywane w trakcie wykonania algorytmu. Zawiera wbudowany system walidacji, który zapobiega wywołaniu symulacji z niewłaściwymi atrybutami klasy Solver. Zawiera funkcjonalność pozwalającą zapisać wyniki symulacji do pliku .csv. Pozwala przeprowadzić symulowane wyżarzanie dla zadanych parametrów początkowych.
   Parametry przyjmowane przez Solver:
   cost: funkcja celu/koszt rozwiązania, minimalizowana
   sol_gen: generator losowego sąsiada
   cool: funkcją implementująca schemat chłodzenia
   probability: funkcja prawdopodobieństwa
   init_sol: rozwiązanie początkowe(generowane poprzez inny, dedykowany algorytm)
   init_temp: temperatura początkowa
   max_iterations: maksymalna ilość iteracji, po której algorytm kończy działanie
   csv_file_path: ścieżka do pliku csv, umożliwia zapis numerycznych wyników symulacji do pliku

2. **Funkcja kosztu:** oblicza koszt na podstawie macierzy przypisania. Sumuje koszty pracy pracowników, maszyn dodaje do tego czas wykonania zlecenia z ustalonym współczynnikiem, a także dolicza karę za przekroczenie zakładanego czasu wykonania zlecenia jeśli przekroczenie jest obecne.

3. **Generator losowego sąsiada:** zwraca zmienioną kopię rozwiązania bazowego, wykonuje jeden “krok” na kopii tyle razy ile wynosi średnica przedziału sąsiedztwa. Jeden krok to próba dopisania lub odjęcia zmiany w trakcie dnia pracy dla losowego pracownika na losowej maszynie, jeśli się nie powiedzie to schemat przypisania pracowników nie jest zmieniany. Porażka następuje, gdy odjęcie pracownika sprawi, iż dla danej maszyny nie zostanie spełnione wymaganie obowiązkowej produkcji lub dodanie pracownika na maszynę jest niemożliwe, gdyż maszyna jest zapełniona pracownikami albo ten pracownik ma już maksymalną liczbę 5emperatur.

4. **Generator rozwiązania dopuszczalnego:** jest to funkcja, która generuje rozwiązanie początkowe dopuszczalne. Generacja odbywa się podobnie do zasady działań algorytmów rodziny Simpleks. Algorytm kolejno stara spełnić wymagania produkcji dla wszystkich maszyn przydzielając im pracowników o największym doświadczeniu. Kolejność zaspokajania wymagań produkcji dla maszyn odbywa się zgodnie z kolejnością: im gorszy stosunek zysku produkcji maszyny do jej wymagań, tym wcześniej dana maszyna zostanie uwzględniona w przydzielaniu pracowników. Jeśli algorytmowi nie uda się spełnić wymagań produkcji, rozszerza czas wykonania zlecenia o jeden dzień i ponownie stara się znaleźć dopuszczalne rozwiązanie w sposób zachłanny. Jeśli zostanie przekroczona liczba maksymalnych rozszerzeń czasu wykonania, algorytm rzuca wyjątek i kończy swoje działanie.

5. **Schematy chłodzenia:** opracowane zostały 4 schematy chłodzenia: liniowy, wielomianowy, eksponencjalny oraz logarytmiczny. Każdy oferuje wybór temperatury początkowej oraz końcowej, a także działanie w seriach poprzez dobór długości tejże.

6. **Funkcja prawdopodobieństwa:** wykorzystana została standardowa funkcja prawdopodobieństwa zmiany stanu znana z praw termodynamiki, opisana w Siddique, N., & Adeli, H. (2016). „Simulated Annealing, Its Variants and Engineering Applications.” *International Journal on Artificial Intelligence Tools*, 25(6), 1630001-1 (24 pages). DOI: 10.1142/S0218213016300015. Jedyną zmianą jest to, iż funkcja zwraca prawdopodobieństwa przejścia równe 1, jeśli znaleziona zostaje poprawa funkcji celu, zamiast robić to już w kodzie algorytmu.

7. **Wyświetlanie danych:** utworzona przez nas biblioteka pozwala również zilustrować w postaci wykresów zmiany wartości poszczególnych parametrów w trakcie trwania symulacji.

Aplikacja

Aby otworzyć interfejs wywołujący algorytm należy uruchomić plik __main__.pyw

Baza danych powinna być umieszczona w pliku formatu .json i zawierać elementy:

machines – lista elementów o składnikach: **id** (int); **hourly_cost** (float); **hourly_gain** (float); **inventory_nr** (int); **max_workers** (int); **demand** (int,float);

employees – lista elementów o składnikach: **id** (int); **hourly_cost** (float); **hourly_gain** (lista mapująca id maszyny na doświadczenie pracownika (float)); **name** (str); **surname** (str); **shift_duartion** (int);

time_span – lista elementów o składnikach: **datetime** (datetime.datetime); **id** (int);

Przykładowy plik json:

```
{
  „machines”: [
    {„id”: 0, „hourly_cost”: 50.0, „hourly_gain”: 120.0, „inventory_nr”:
100, „max_workers”: 2, „demand”: 10000},
    {„id”: 1, „hourly_cost”: 57.6, „hourly_gain”: 106.1, „inventory_nr”:
101, „max_workers”: 6, „demand”: 11820}
],

„employees”: [
  {„id”: 0, „hourly_cost”: 15, „hourly_gain”: {„0”: 20, „1”: 15}, „name”:
„Ana”, „surname”: „Doe”, „shift_duration”: 4},
  {„id”: 1, „hourly_cost”: 25, „hourly_gain”: {„0”: 20, „1”: 15}, „name”:
„John”, „surname”: „Smith”, „shift_duration”: 8}
],

„time_span”: [
  {„datetime”:”2023-11-01T06:00:00”,”id”:0},
  {„datetime”:”2023-11-01T07:00:00”,”id”:1}
]
}
```

Po uruchomieniu ukazuje się okno z parametrami do uzupełnienia:

Initial 8emperaturę:

temperatura początkowa – liczba nieujemna;

Final 8emperaturę:

temperatura końcowa – nieujemna oraz nie większa niż początkowa;

Number of iterations:

maksymalna liczba iteracji – liczba całkowita dodatnia;

Iterations in single 8emperaturę:

liczba iteracji w jednej temperaturze – liczba całkowita dodatnia i nie większa niż maksymalna liczba iteracji;

Cooling function:

sposób chłodzenia, do wyboru: liniowe, wielomianowe, eksponencjalne, logarytmiczne;

Degree of polynomial:

w przypadku wyboru chłodzenia wielomianowego jest to stopień wielomianu, liczba całkowita dodatnia

Neighbor diameter:

rozmiar obszaru sąsiedztwa: liczba całkowita dodatnia

W przypadku próby wpisania wartości niezgodnych z powyższymi wytycznymi interfejs usunie wpisaną wartość.

Jeśli wszystkie dane są wypełnione można zobrazować funkcję chłodzenia za pomocą przycisku **‘’Plot Cooling”**.

Aby zaimportować bazę danych należy nacisnąć przycisk **''Browse file''** i wybrać plik o rozszerzeniu json. Po wybraniu pliku wyświetlony zostanie komunikat o poprawnym zaimportowaniu danych, bądź o wybraniu pliku o złym rozszerzeniu.

Jeśli baza danych została wczytana i parametry zostały uzupełnione, można uruchomić algorytm naciskając przycisk **''Run algorithm''**.

Po zakończeniu działania algorytmu otworzy się okno zawierające 4 wykresy:

- przebieg temperatury w kolejnych iteracjach;

- prawdopodobieństwo wyboru gorszego rozwiązania;

- dotychczasowy najlepszy koszt rozwiązania;

- koszt rozwiązania w kolejnych iteracjach.

oraz 3 wartości:

- ostateczny koszt najlepszego rozwiązania;

- bezwzględna poprawa rozwiązania;

- względna poprawa rozwiązania.

Dodatkowo możliwe jest zapisanie rozwiązania w postaci listy macierzy reprezentujących przydzielenie pracowników. Każda macierz odpowiada jednemu przedziałowi czasu. Aby zapisać należy nacisnąć przycisk **''Save solution''**. Możliwe formaty plików to json i txt. Jeśli jakiś program ma później korzystać z tych danych to lepiej wybrać json, jeśli plik ma służyć jedynie podglądowi to plik txt jest bardziej czytelny dla człowieka.

Testy

Wykonano testy sprawdzające zachowanie algorytmu dla wprowadzanych parametrów:

- schemat chłodzenia,

- rozwiązanie początkowe,

- maksymalna liczba iteracji,

- rozmiar sąsiedztwa,

- uwarunkowanie problemu,

- temperatura początkowa.

Domyślne parametry każdego z testów wynosiły:

schemat chłodzenia – liniowy;

rozwiązanie początkowe – koszt równy $1{,}17694*10^5$;

temperatura początkowa – 30;

temperatura końcowa – 0.1;

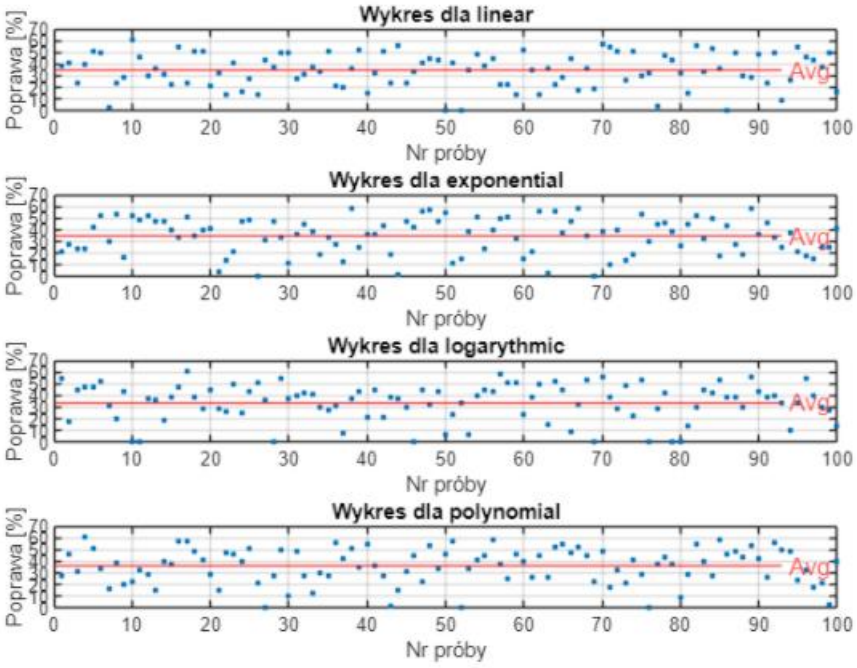
maksymalna liczba iteracji – 1000;

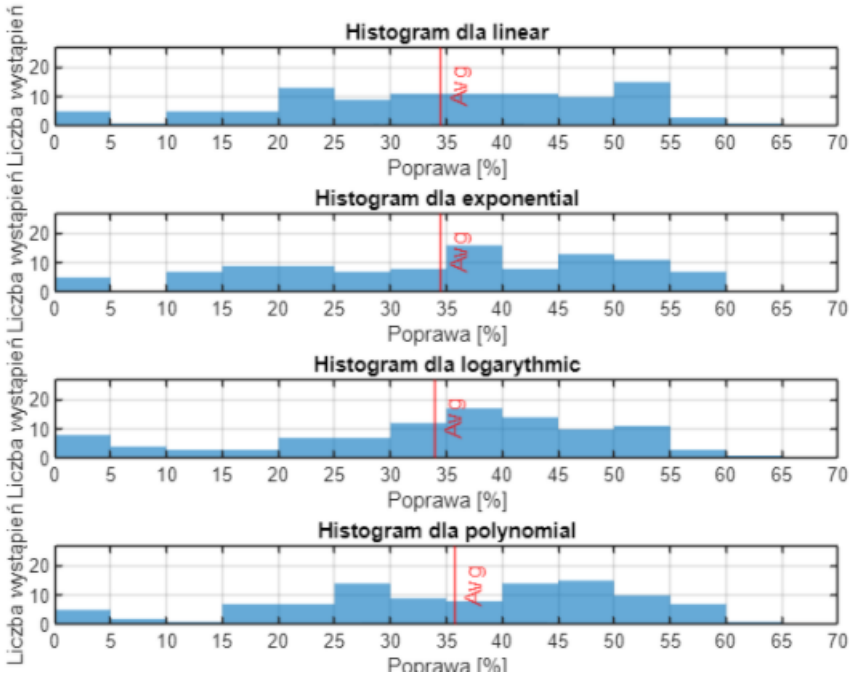
liczba iteracji w jednej temperaturze – 1;

Dla każdego przypadku uruchomiono algorytm 100 razy.

Testy ze względu na schemat chłodzenia

Testując wpływ schematu chłodzenia przeprowadzono testy dla każdej z możliwych funkcji (liniowa, eksponencjalna, logarytmiczna, wielomianowa).



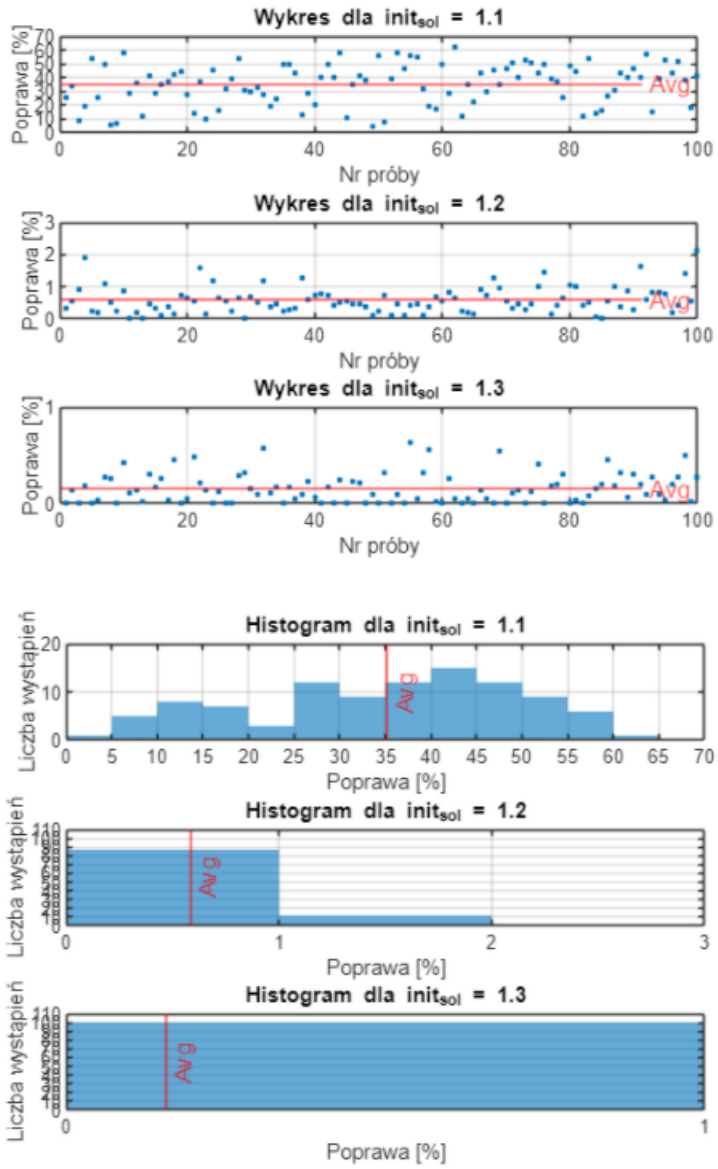


	cooling schedule	avg	std
1	"linear"	34.5297	14.5814
2	"exponential"	34.4949	15.2177
3	"logarythmic"	33.9874	15.8362
4	"polynomial"	35.7570	14.9766

Średnia poprawa jest dla każdego ze schematów podobna i nie wpływa znacząco na wynik.

Testy ze względu na rozwiązanie początkowe

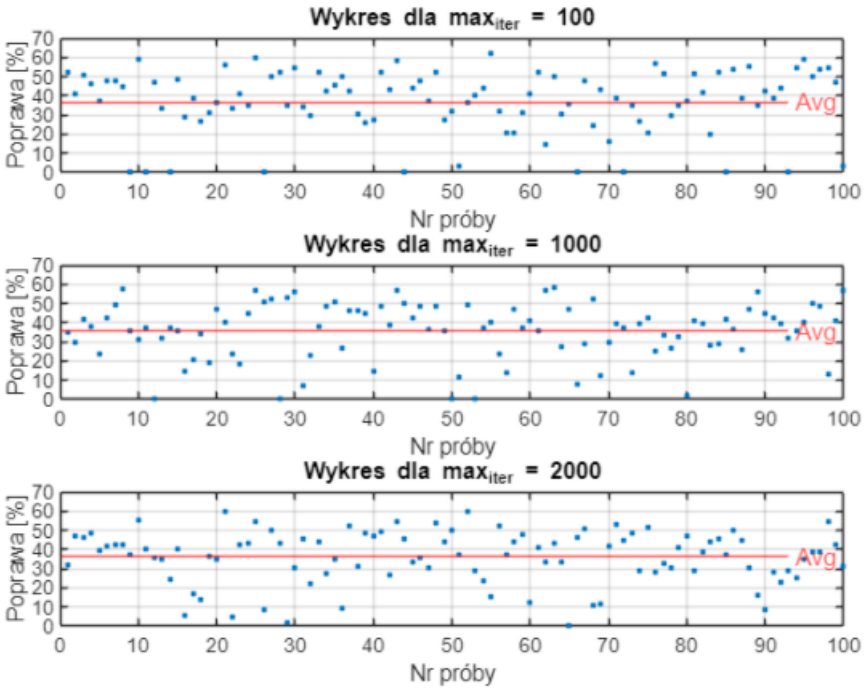
W tym teście użyto trzech baz danych różniących się jedynie zapotrzebowaniem na konkretne towary. Zapotrzebowanie jest najmniejsze dla pierwszego przypadku, później ilości rosną.

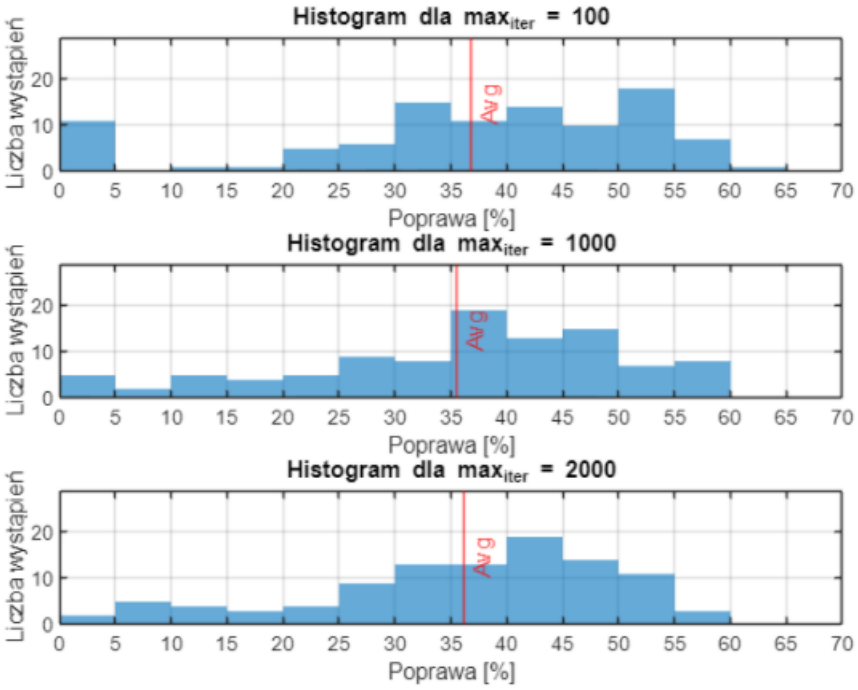


	init_sol	avg	std
1	"1.1"	35.1321	14.6535
2	"1.2"	0.5828	0.4153
3	"1.3"	0.1561	0.1558

Po uruchomieniu algorytmu okazało się, że aby spełnić wymagania co do ilości dla drugiego przypadku trzeba było wydłużyć termin o 8 dni, a dla trzeciego o 23 dni. Oznacza to, że wymagania były zbyt duże i algorytm nie jest w stanie znaleźć terminowego rozwiązania. Dlatego koszt już na początku jest duży z powodu kary za niespełniony termin, a kolejne iteracje nie są w stanie wiele wnieść do ostatecznego rozwiązania.

Testy ze względu na maksymalną liczbę iteracji



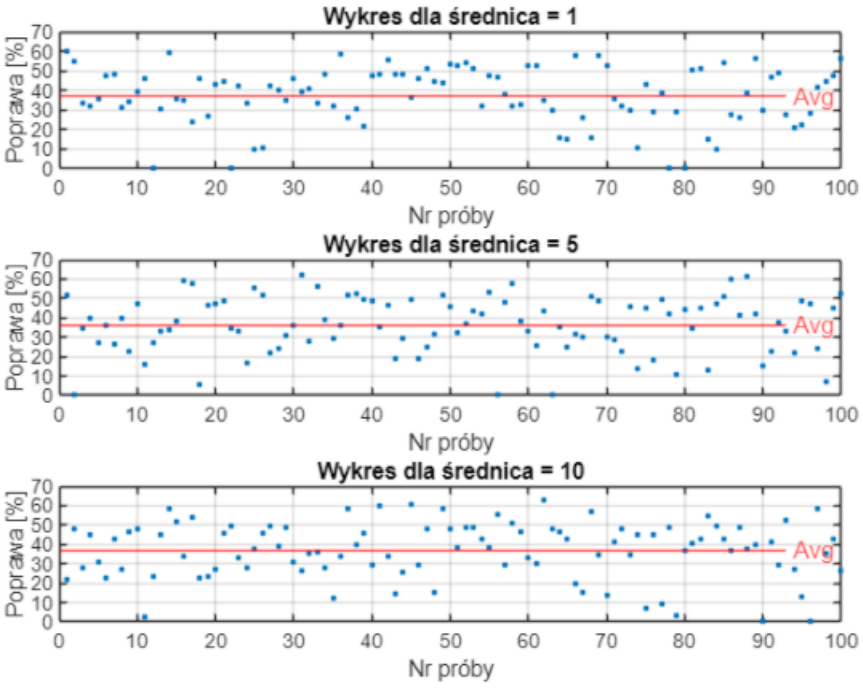


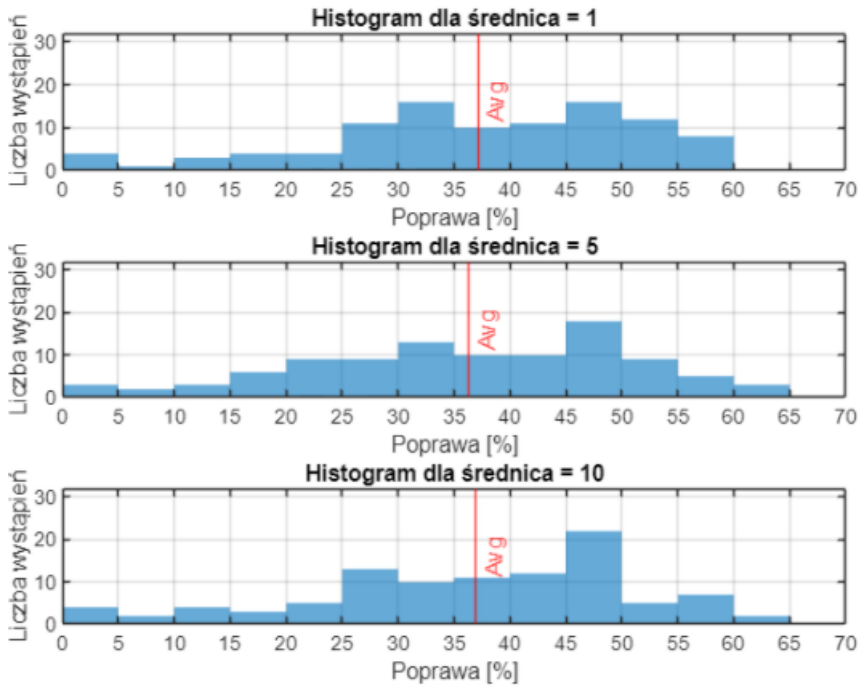
	max_iter	avg	std
1	100	36.8685	16.5803
2	1000	35.5326	14.7207
3	2000	36.1331	13.7685

Dla każdej z testowanych iteracji średnia poprawa była podobna, jednak odchylenie standardowe układa się malejąca w stosunku do rosnącej liczby iteracji. Można stąd wywnioskować, że zwiększając liczbę iteracji otrzymamy mniej rozbieżne wyniki. Wiąże się to natomiast z czasem trwania działania algorytmu.

Testy ze względu na rozmiar sąsiedztwa

Rozmiar sąsiedztwa określa ilość podkroków wykonywanych podczas tworzenia sąsiedniego rozwiązania. Im jest większa tym bardziej rozwiązanie sąsiednie różni się od obecnego.



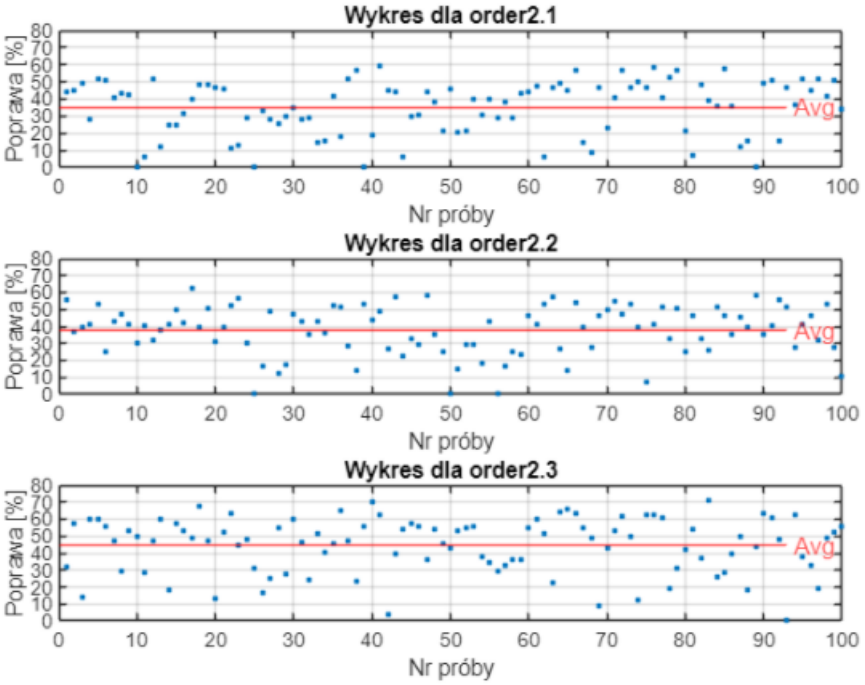


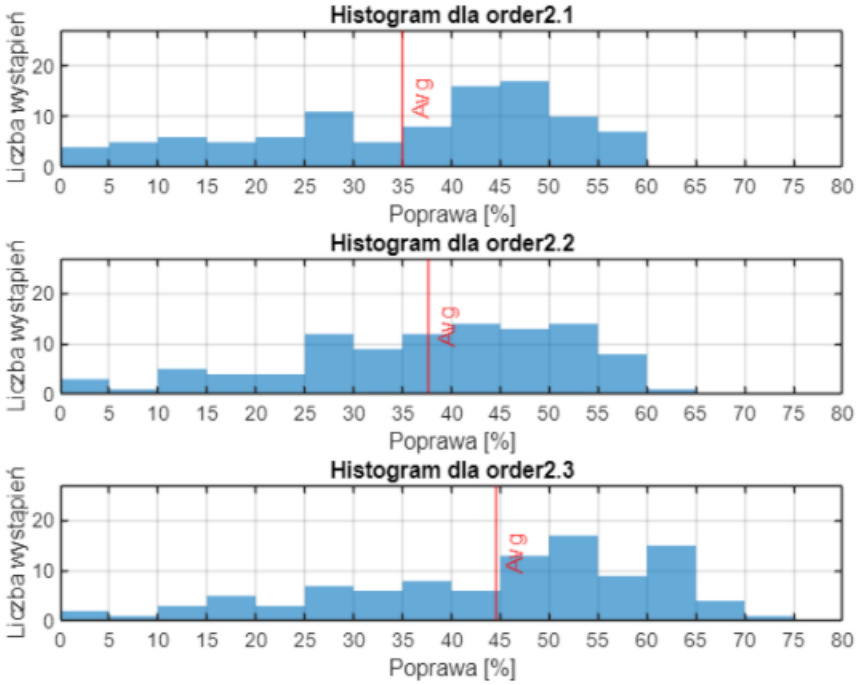
	diameter	avg	std
1	1	37.1516	14.5487
2	5	36.3342	14.5311
3	10	36.9548	14.5926

Z analizy wynika, że rozmiar sąsiedztwa nie ma w tym problemie istotnego znaczenia.

Testy ze względu na uwarunkowanie problemu

W tym teście skorzystano z trzech różnych baz danych aby zobaczyć działanie algorytmu w różnych przypadkach. W żadnym przypadku nie doszło do przekroczenia czasu wykonania zlecenia.



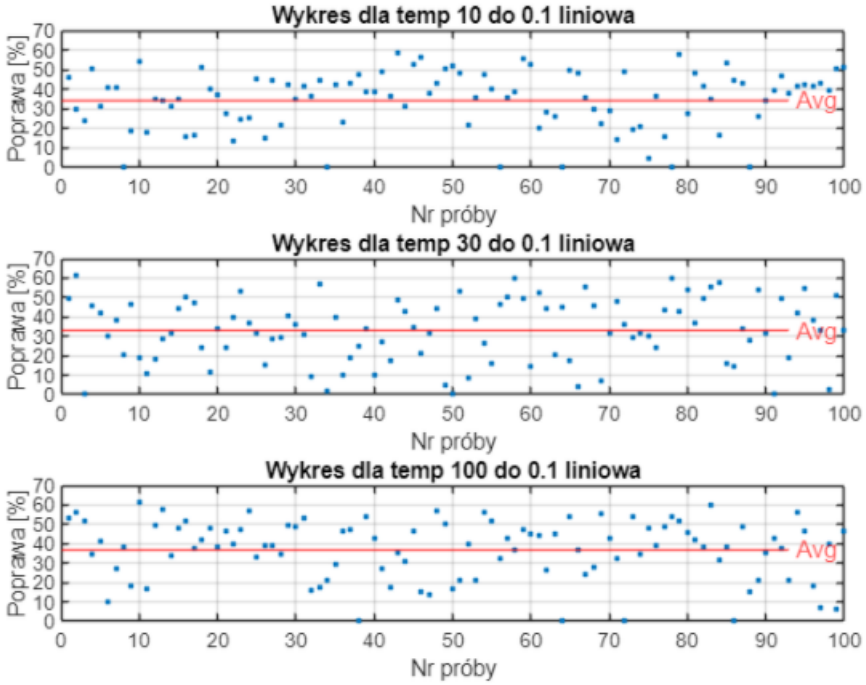


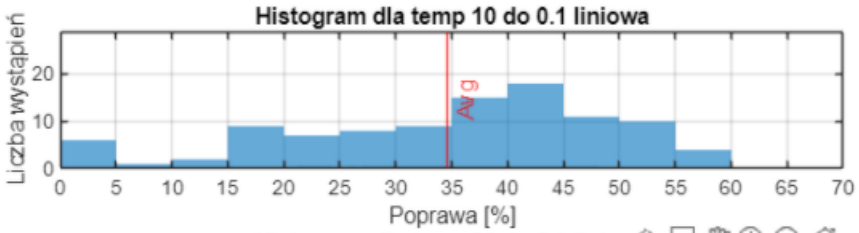
	Order	avg	std
1	"1.1"	35.0215	15.8485
2	"1.2"	37.6335	14.3462
3	"1.3"	44.6214	16.2370

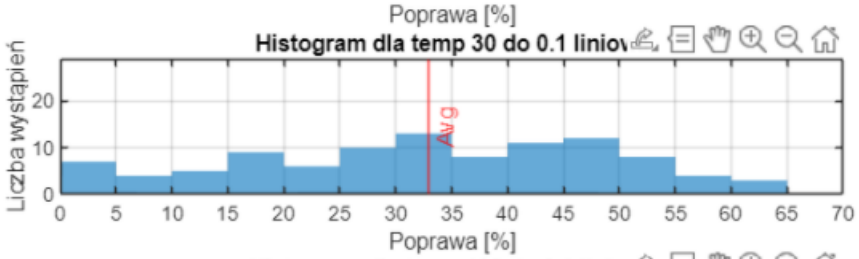
W tym przypadku widać, że dla różnego zestawu danych można uzyskać znaczącą różnicę w poprawie.

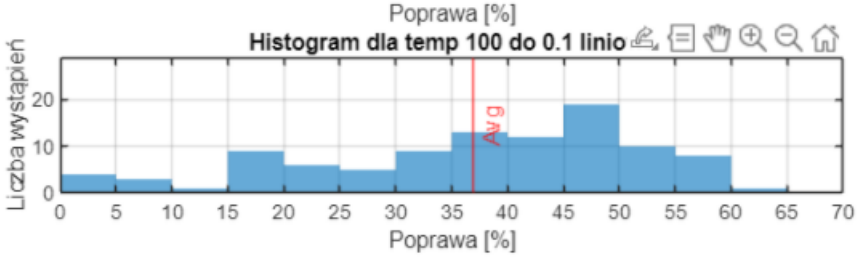
Testy ze względu na temperaturę początkową

Testy przeprowadzono dla 4 schematów chłodzenia, dla każdego ustawiano temperaturę początkową kolejno 10, 30, 100.

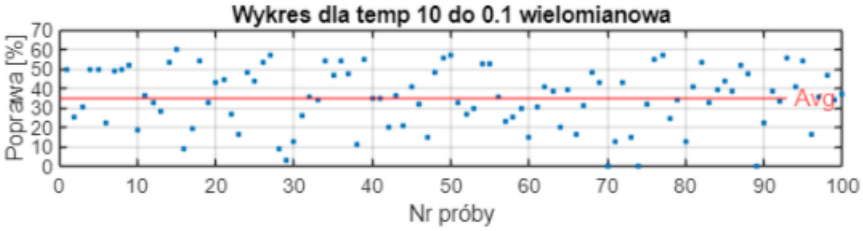


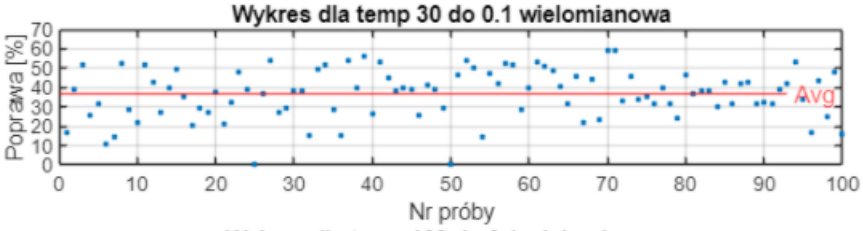


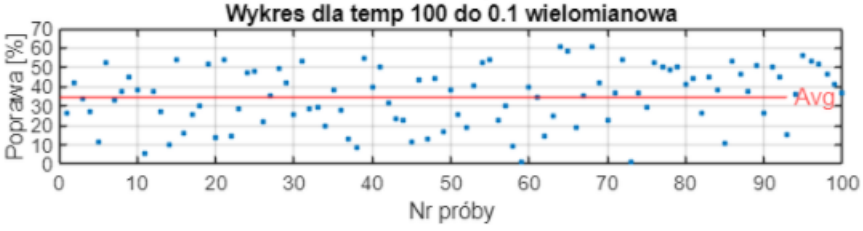


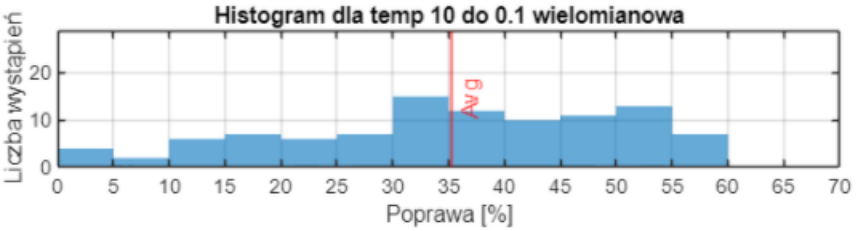


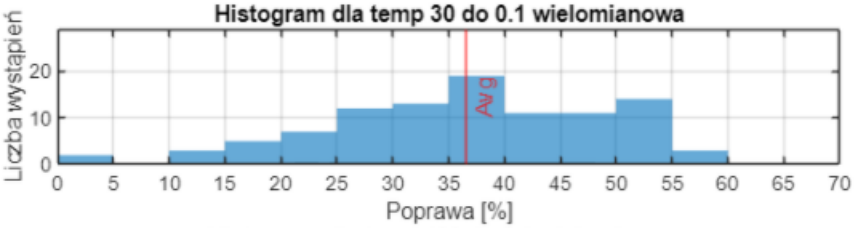
	temp	avg	std
1	"10 to 0.1"	34.6164	14.6530
2	"30 to 0.1"	32.8908	16.2445
3	"100 to 0.1"	36.9957	15.3011

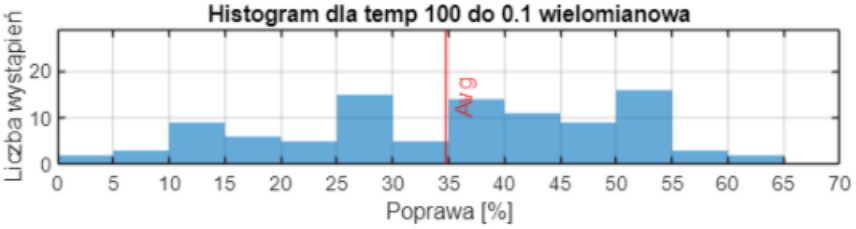




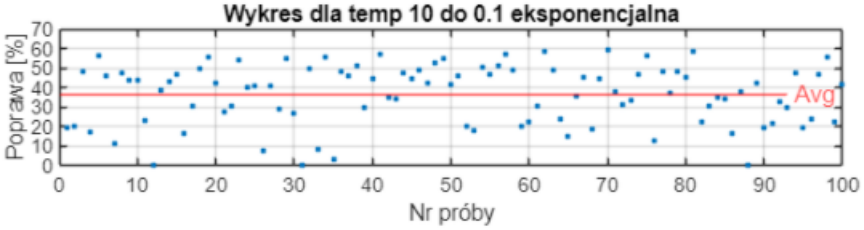


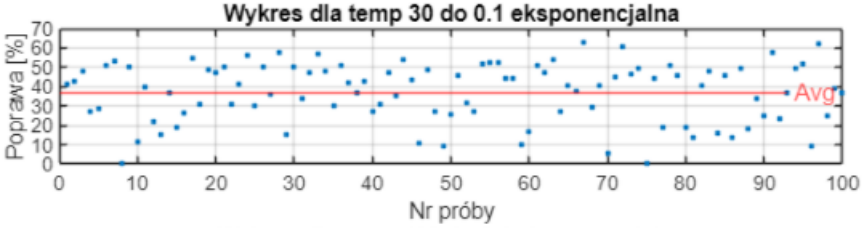


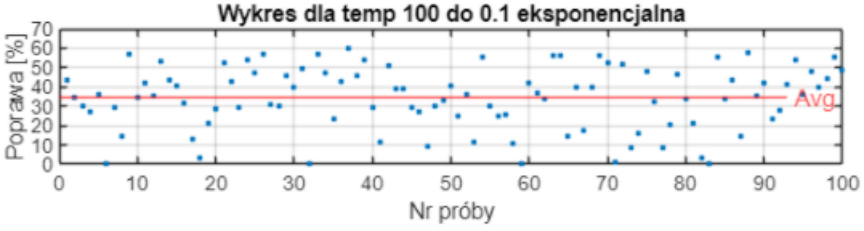


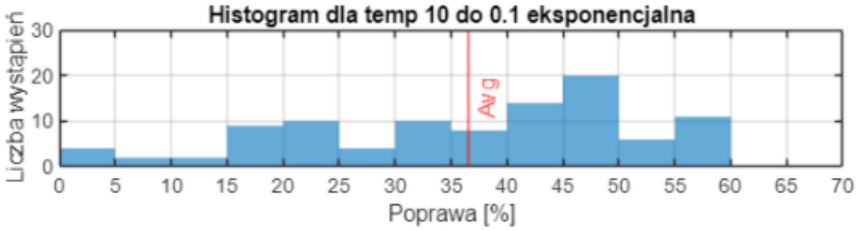


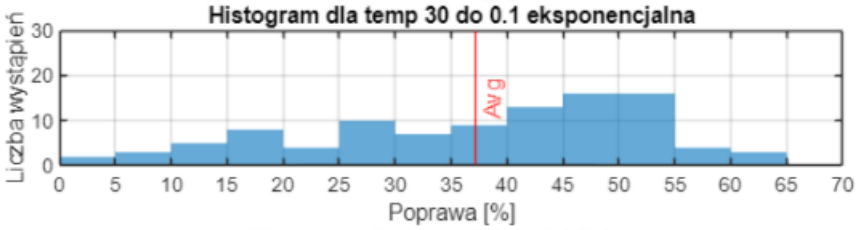
	temp	avg	std
1	"10 to 0.1"	35.2768	15.2035
2	"30 to 0.1"	36.5438	12.6170
3	"100 to 0.1"	34.7890	15.0441

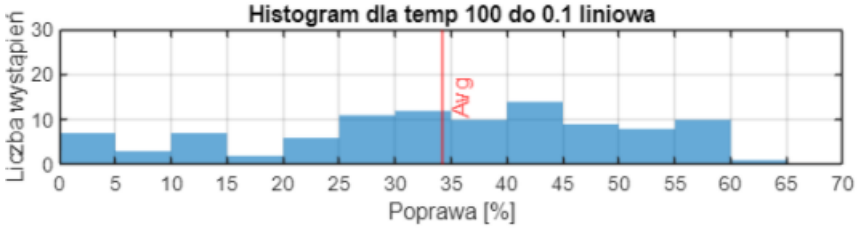




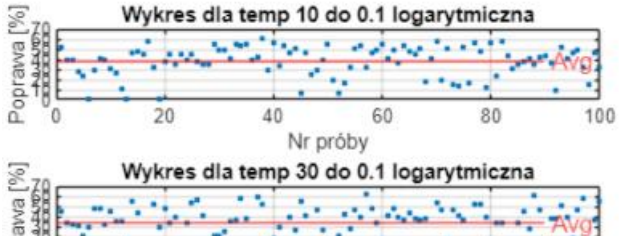


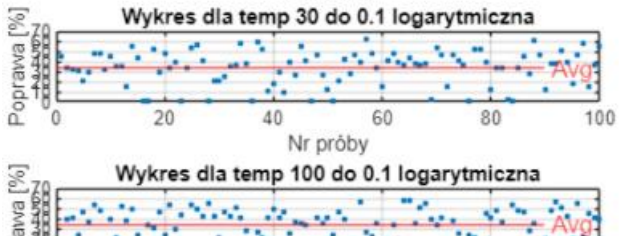


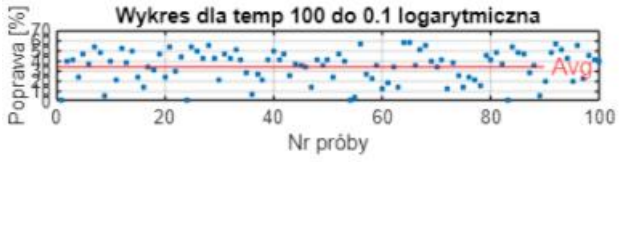


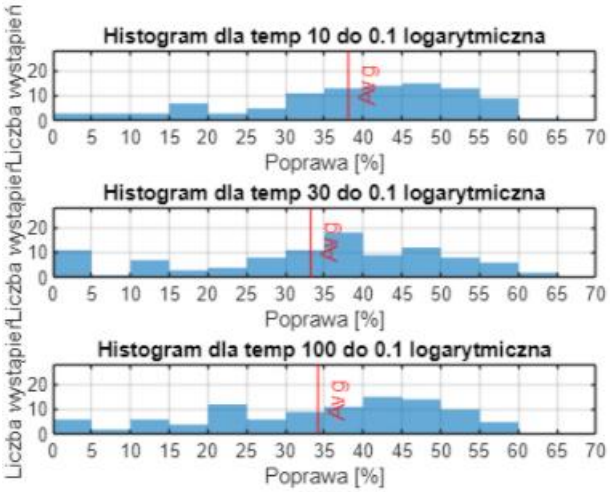


	temp	avg	std
1	"10 to 0.1"	36.5285	15.1885
2	"30 to 0.1"	37.1357	15.1414
3	"100 to 0.1"	34.2716	16.1198









	temp	avg	std
1	"10 to 0.1"	38.0783	14.8013
2	"30 to 0.1"	33.3150	16.9027
3	"100 to 0.1"	34.1859	15.2887

Im większa temperatura tym większa szansa na przyjęcie gorszego rozwiązania. Jednak po wykonaniu testów trudno zauważyć różnicę w średniej poprawie. Dla każdego schematu temperatura miała inny wpływ. Największą różnice można było zaobserwować dla zmian w schemacie liniowym i logarytmicznym, a w wielomianowym i eksponencjalnym wpływ temperatury początkowej był niewielki.

Wnioski

Po przeprowadzeniu testów dla różnych parametrów zauważyliśmy, że parametry związane z chłodzeniem nie mają bardzo istotnego wpływu na poprawę działania algorytmu. Największą różnice zaobserwowano w przypadku testowania wpływu rozwiązania początkowego. W przypadku gdy termin jest zbyt krótki w stosunku do zapotrzebowania na produkt poprawa algorytmu jest zazwyczaj niewielka lub żadna. Ogólnie dla różnych baz danych algorytm działa poprawnie.

Jeśli chodzi o rozszerzenie aplikacji, można w niej dodać cofanie się w okolice dotychczasowego najlepszego rozwiązania w przypadku niewielkiej poprawy i tworzenie kolejnych sąsiadów.