Jakub Kasperski

Sprawozdanie

Symulowane wyżarzanie

Informacje techniczne o użytym sprzęcie oraz oprogramowaniu:

- Komputer z systemem Windows 10 x64
- Inter(R) Core(TM) i7-4790K @4.00GHz
- 16 GB RAM
- Użyte środowisko: Jupyter notebook

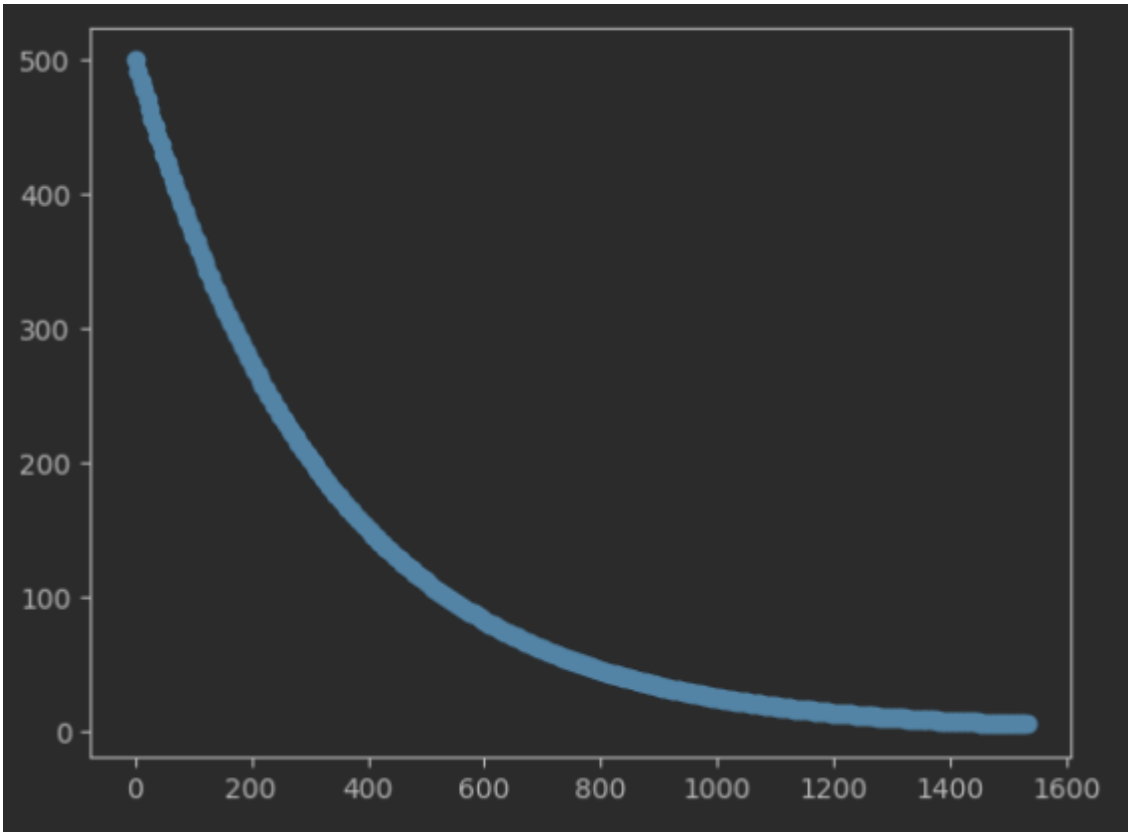
Ćwiczenie zrealizowano przy użyciu interpretera Python 3, z wykorzystaniem bibliotek: matplotlib, numpy, random, PIL oraz math

Zadanie 1

1.1 – dla losowo wygenerowanych 30 punktów

a) temperatura początkowa = 500
   temperatura końcowa = 5
   funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
   sposób zamiany punktów -> zamień losowe punkty

   funkcja temperatury:



Rysunek 1
(wszystkie funkcje temperatury, które maleją wykładniczo wyglądają identycznie, więc nie będę wstawiał ich zdjęć)

Wizualizacja :

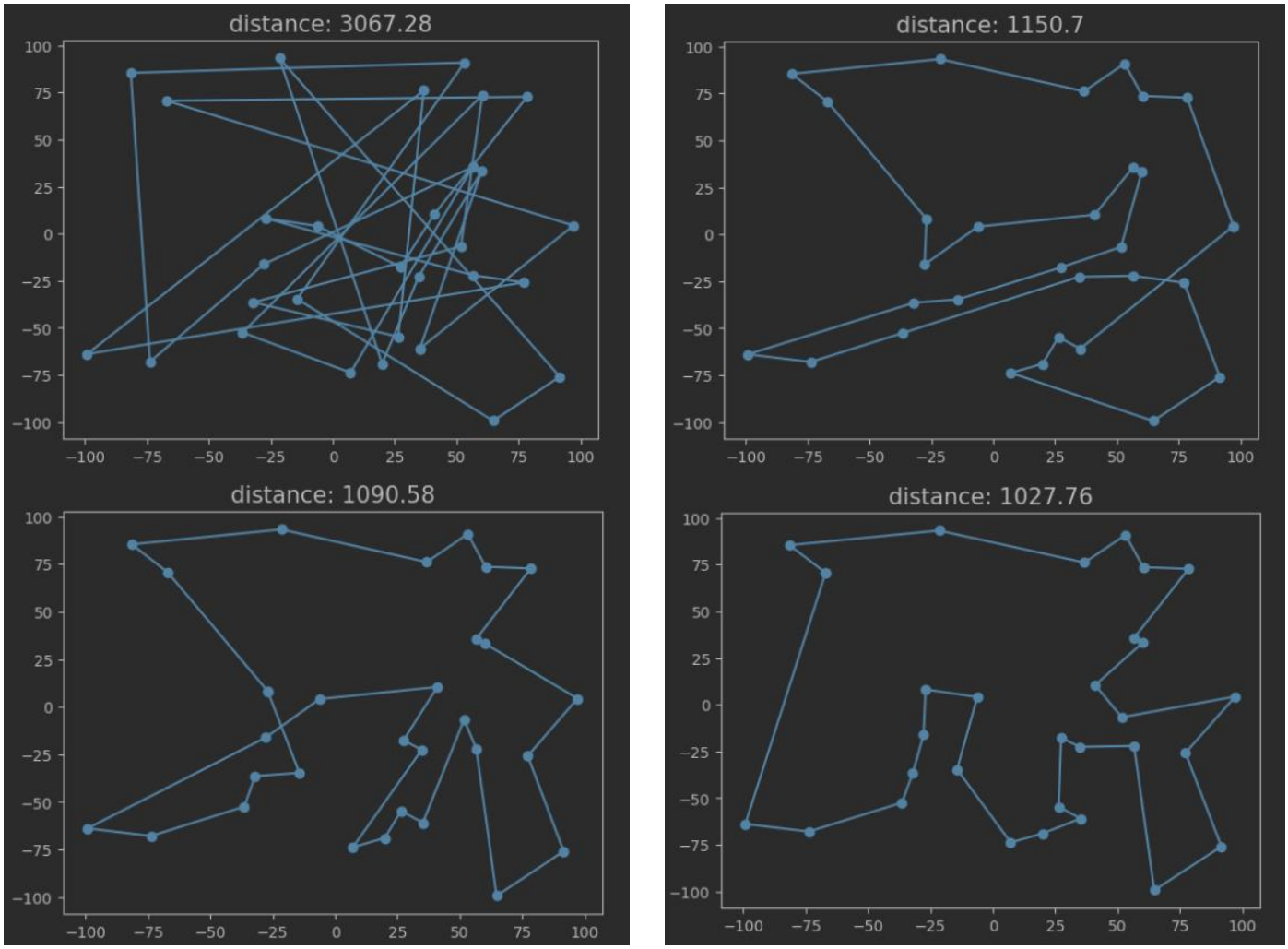
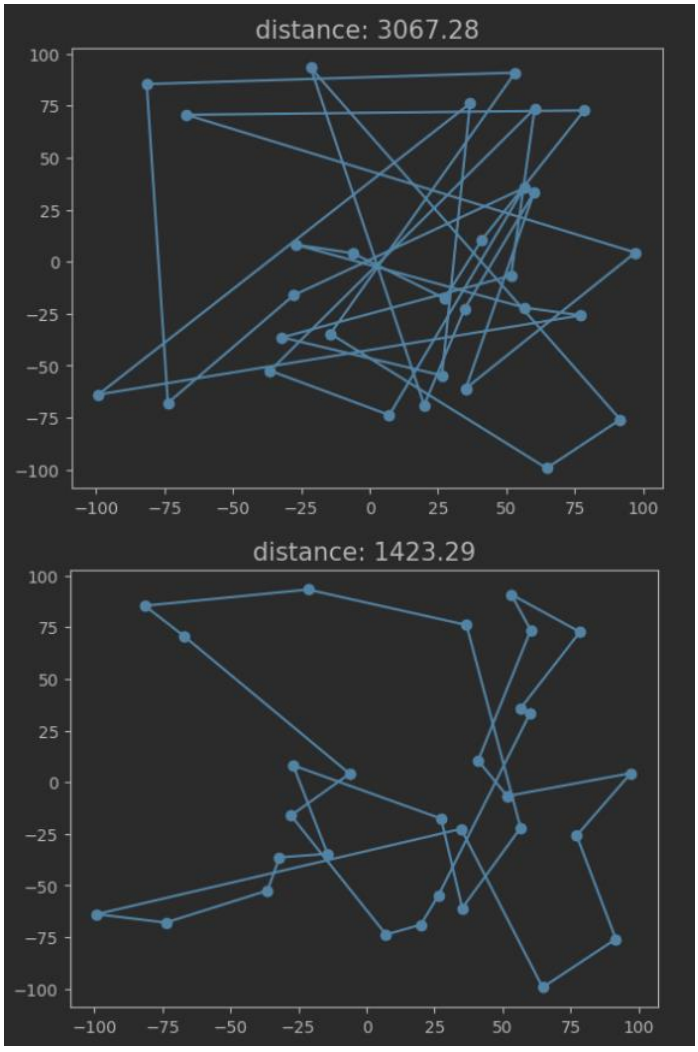


Tabela 1

b) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
sposób zamiany punktów -> zamień sąsiednie punkty

Wizualizacja :



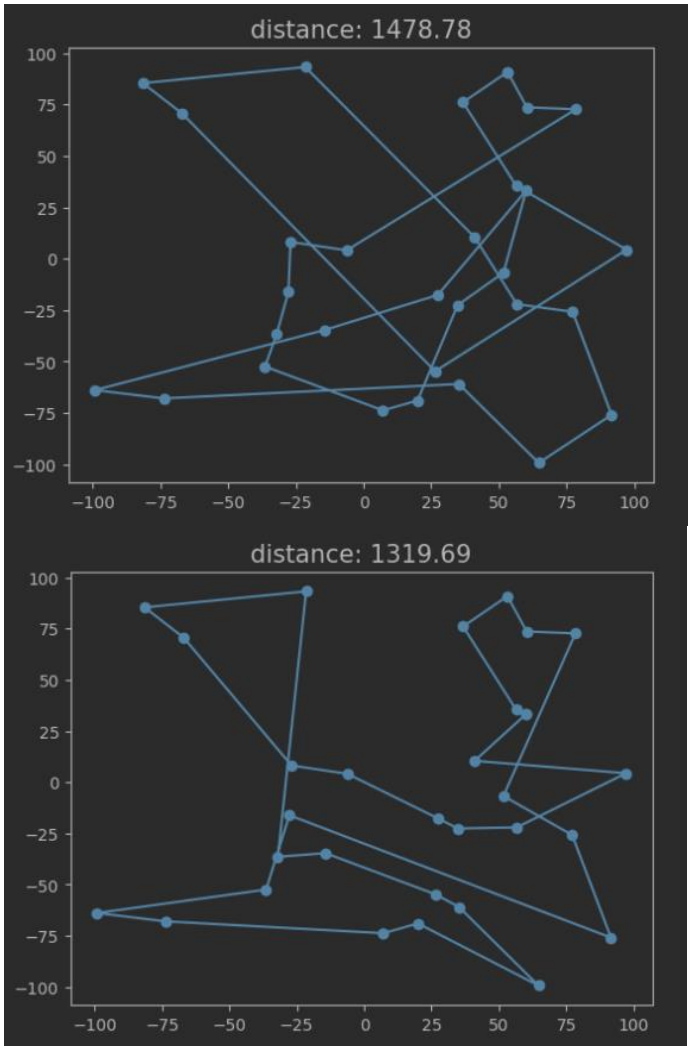


Tabela 2

c) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
sposób zamiany punktów ->
80% na zamianę losowych
20% na zamianę sąsiednich

Wizualizacja :

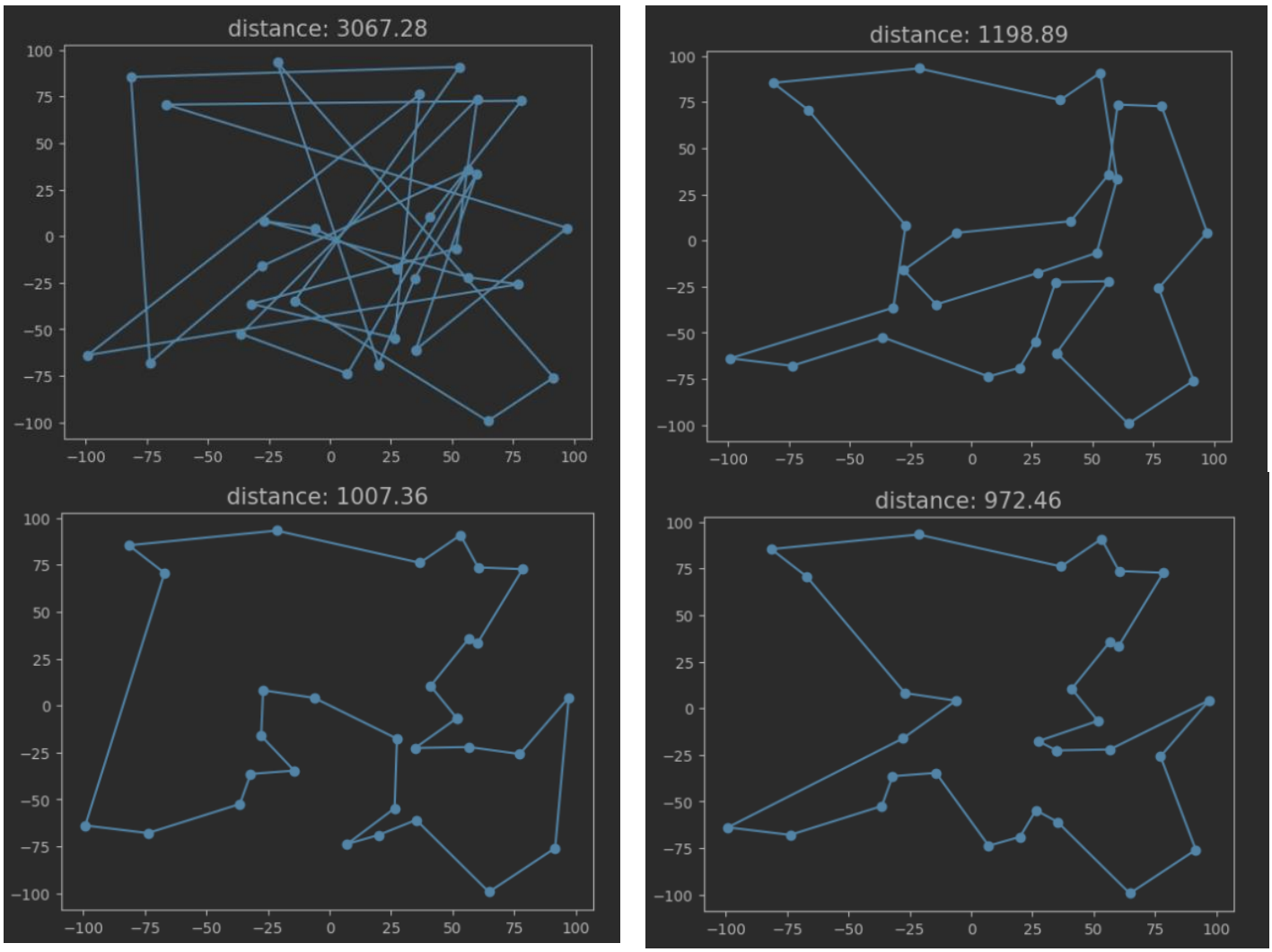


Tabela 3

1.2 – dla losowo wygenerowanych 100 punktów

a) temperatura początkowa = 500
   temperatura końcowa = 5
   funkcja spadku temperatury -> $t_n = 0.9997\ t_{n-1}$
   sposób zamiany punktów -> zamień losowe punkty

   Wizualizacja :

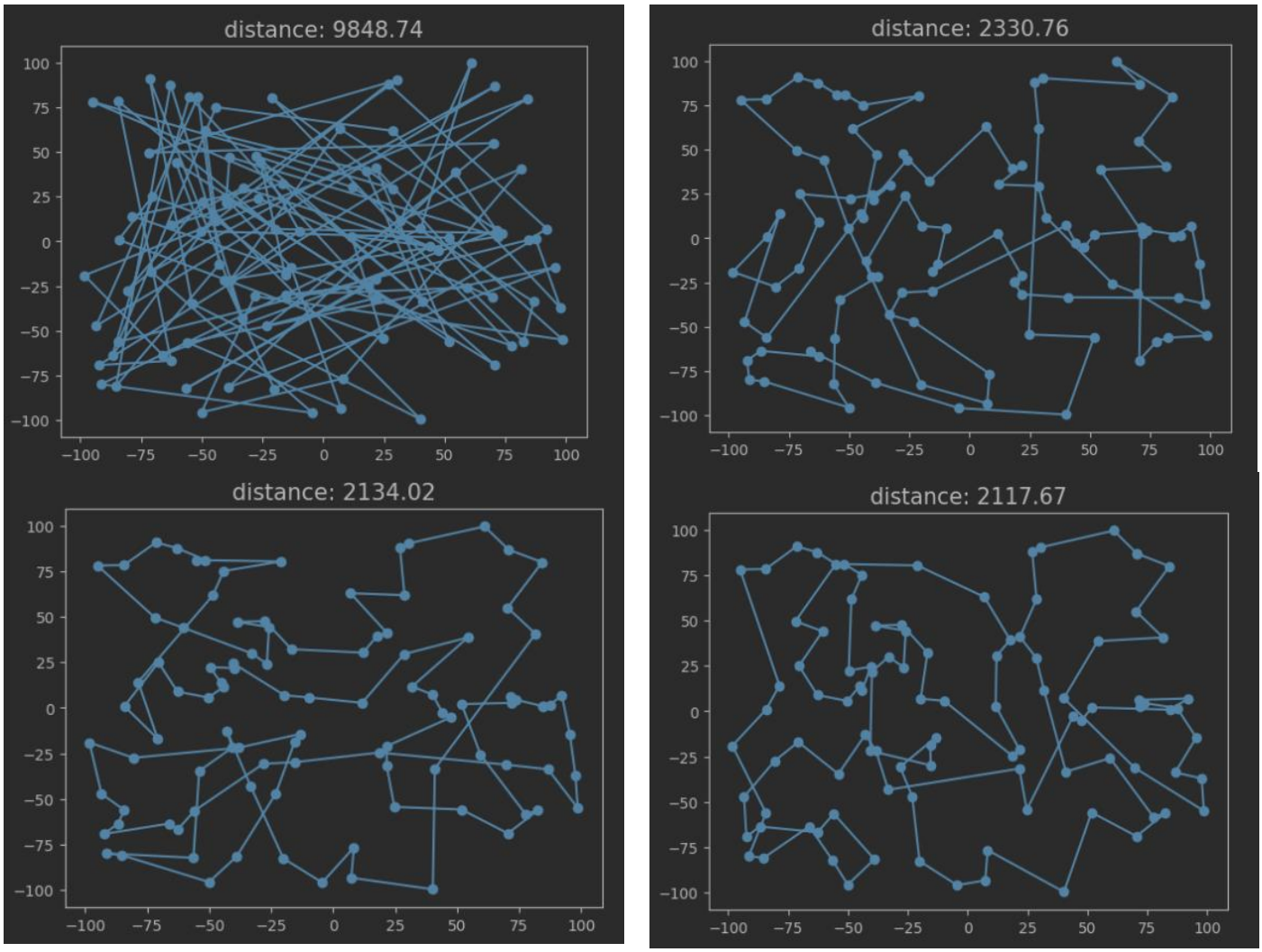


Tabela 4

b) temperatura początkowa = 500
temperatura końcowa = 1
funkcja spadku temperatury -> $t_n = 0.999\ t_{n-1}$
sposób zamiany punktów -> zamień sąsiednie punkty

Wizualizacja :

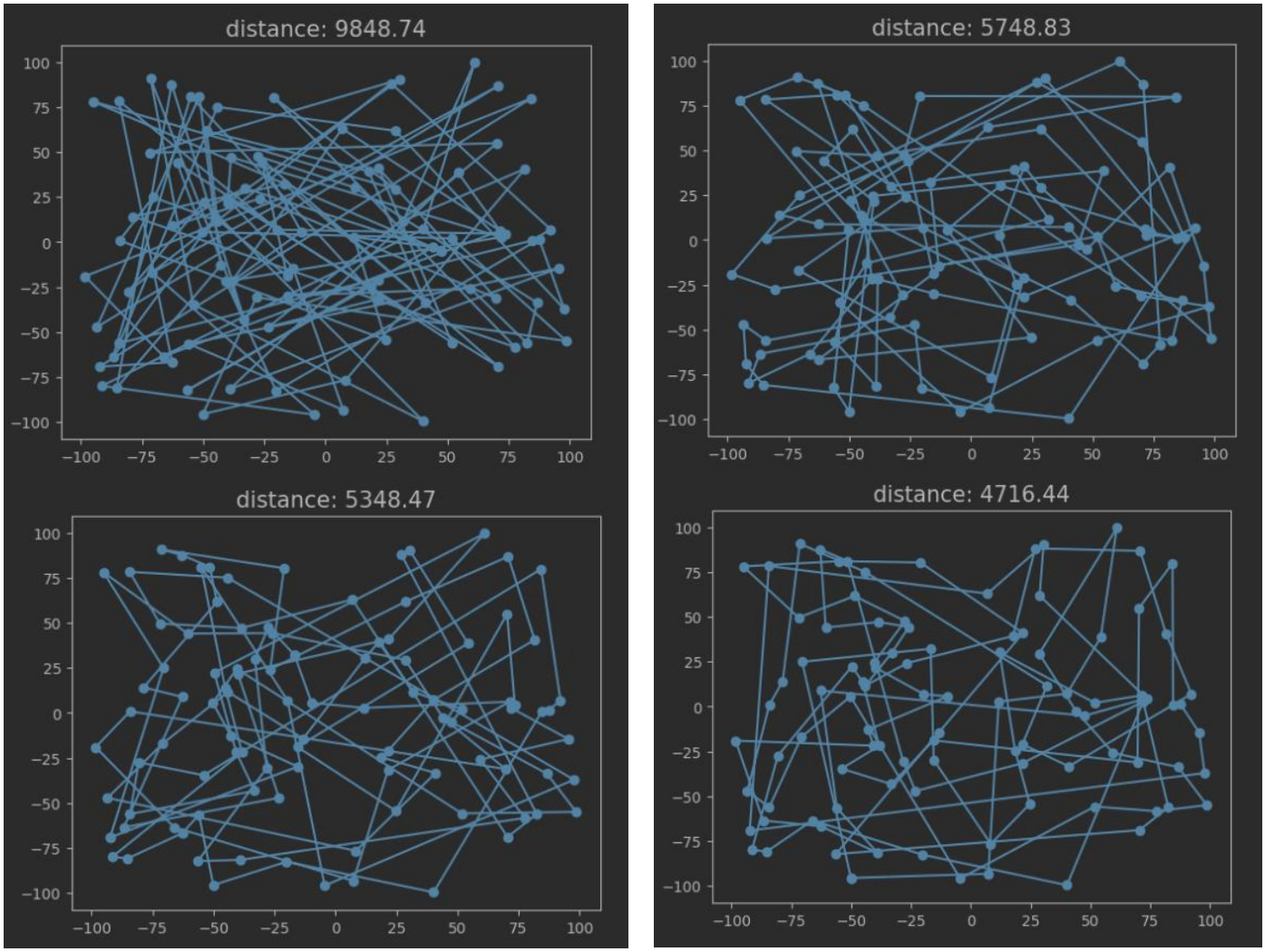


Tabela 5

c) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
sposób zamiany punktów ->
80% na zamianę losowych
20% na zamianę sąsiednich

Wizualizacja :

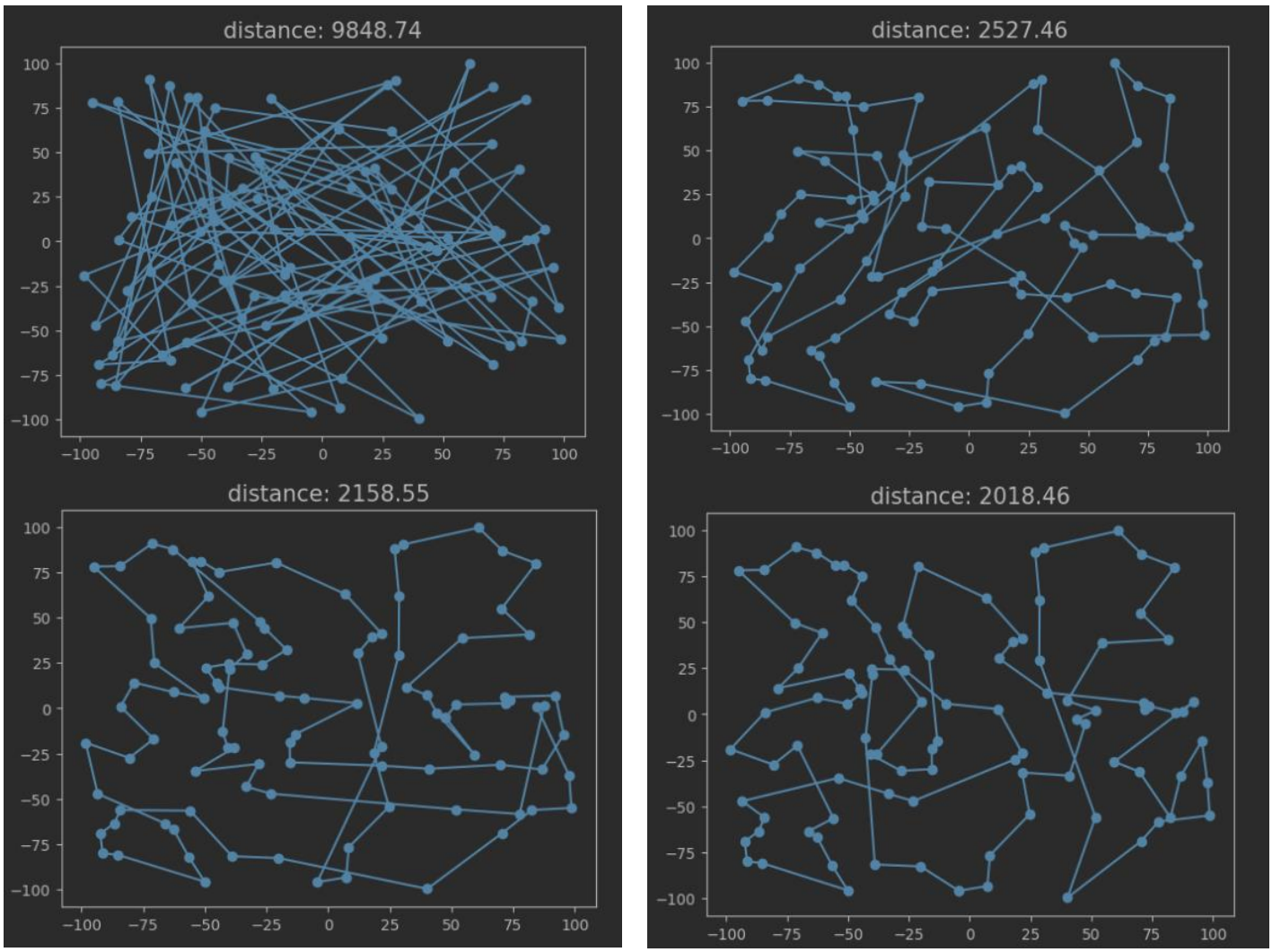


Tabela 6

1.3 – dla czterech grup punktów (łącznie 30 punktów)

a) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
sposób zamiany punktów -> zamień losowe punkty

Wizualizacja :

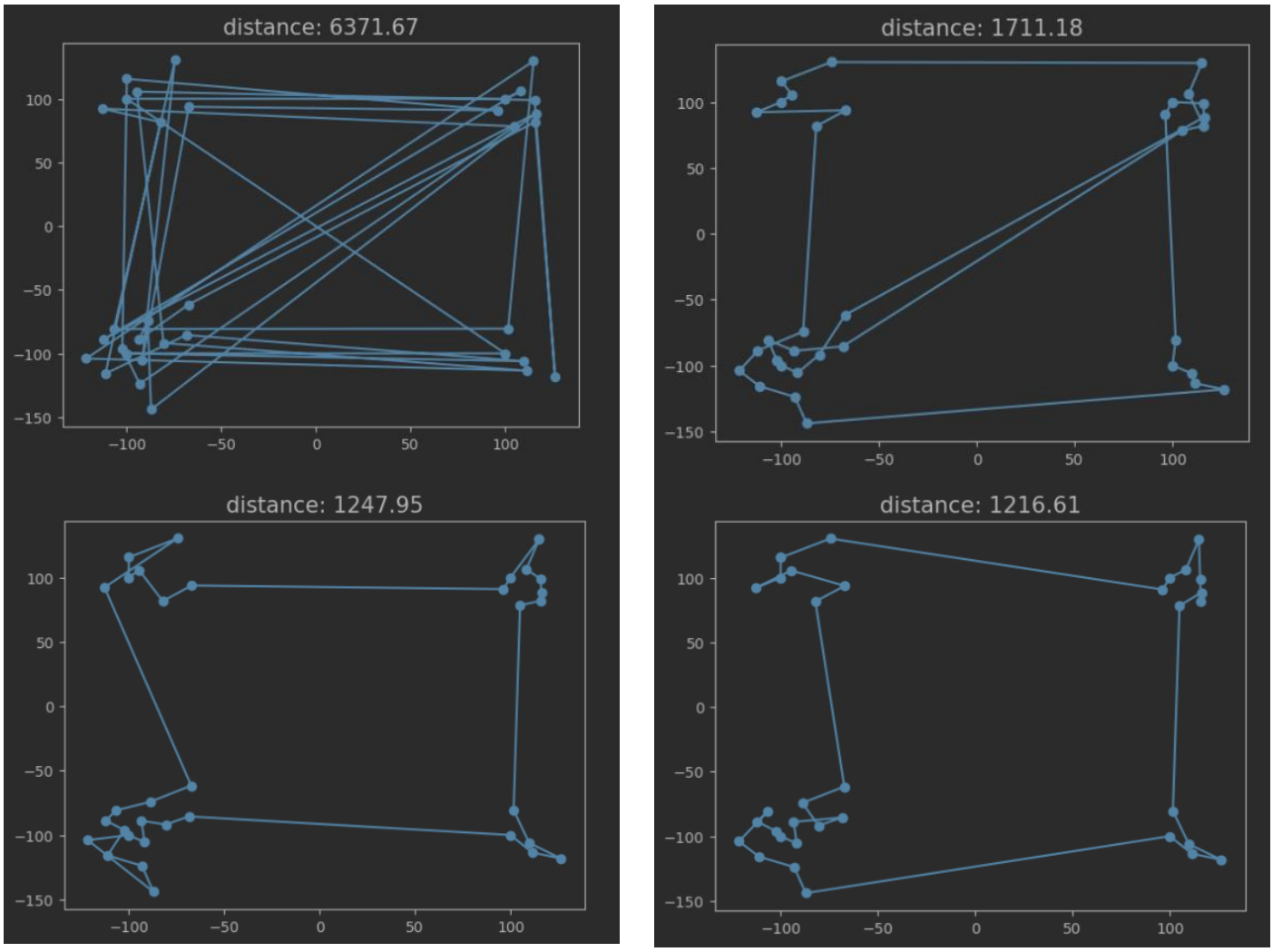


Tabela 7

b) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
sposób zamiany punktów -> zamień sąsiednie punkty

Wizualizacja :

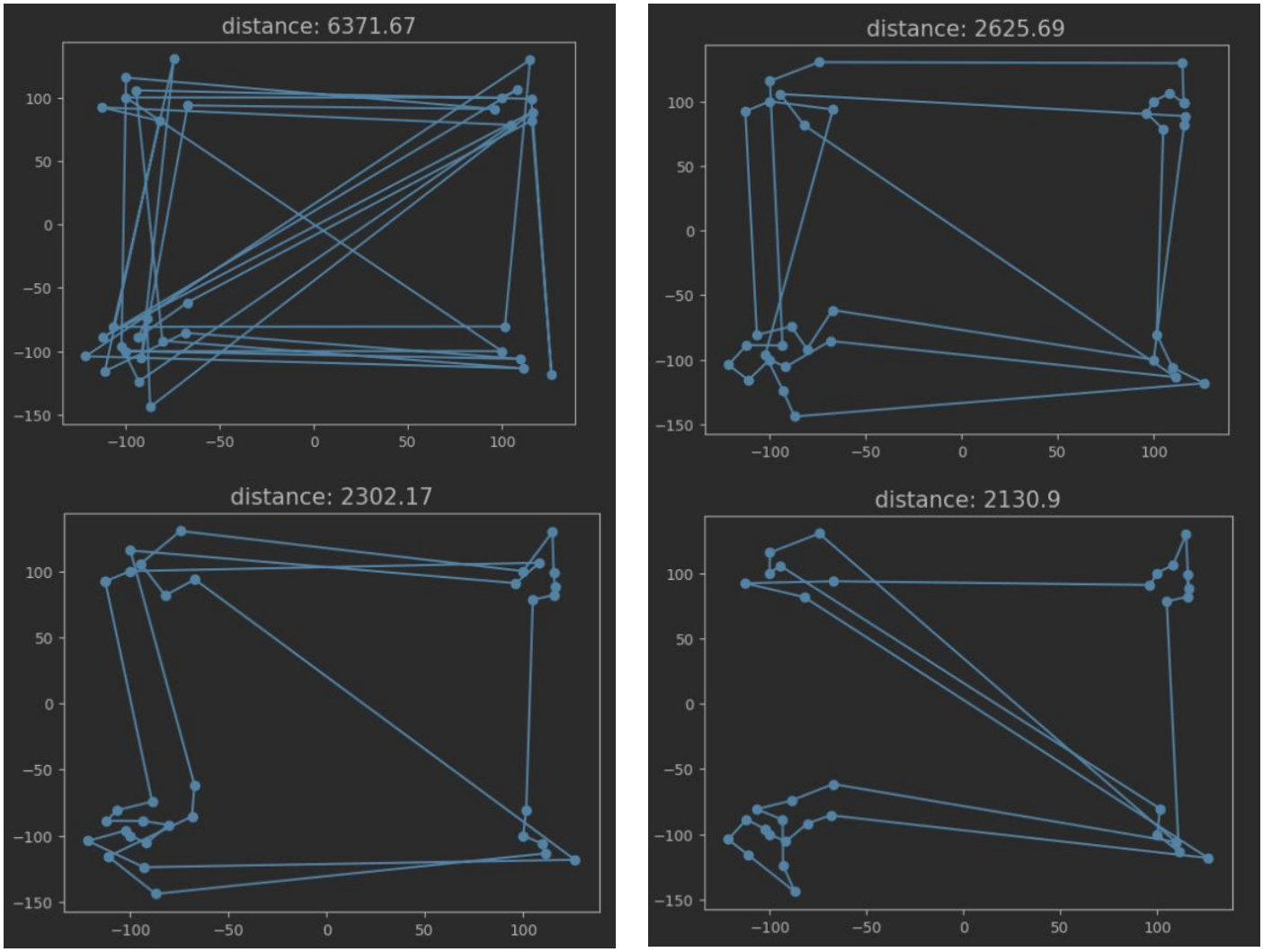


Tabela 8

c) temperatura początkowa = 500

temperatura końcowa = 5

funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$

sposób zamiany punktów ->

80% na zamianę losowych

20% na zamianę sąsiednich

Wizualizacja :

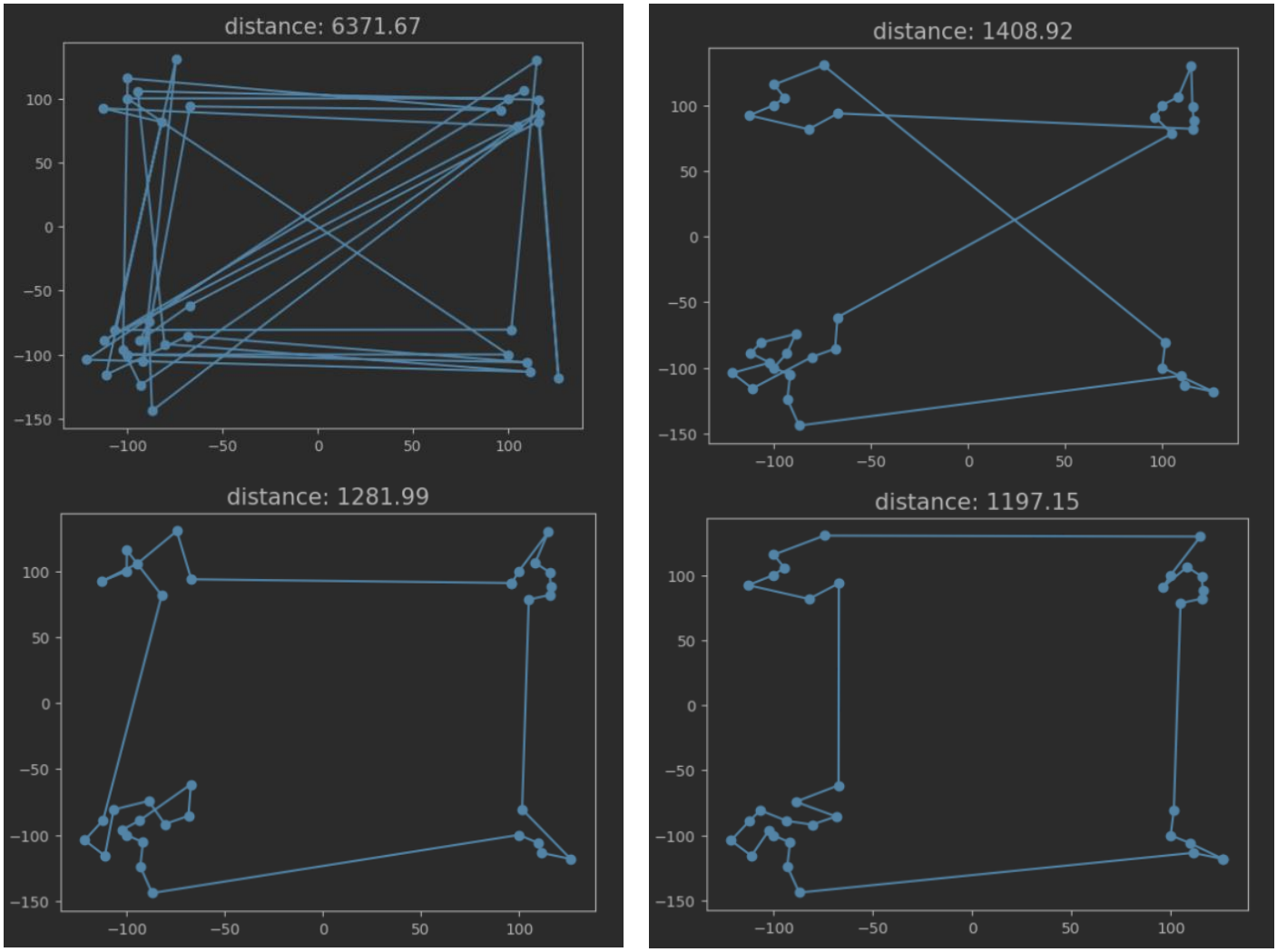


Tabela 9

1.4 – dla dziewięciu grup punktów (łącznie 50 punktów)

a) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
sposób zamiany punktów -> zamień losowe punkty

Wizualizacja :

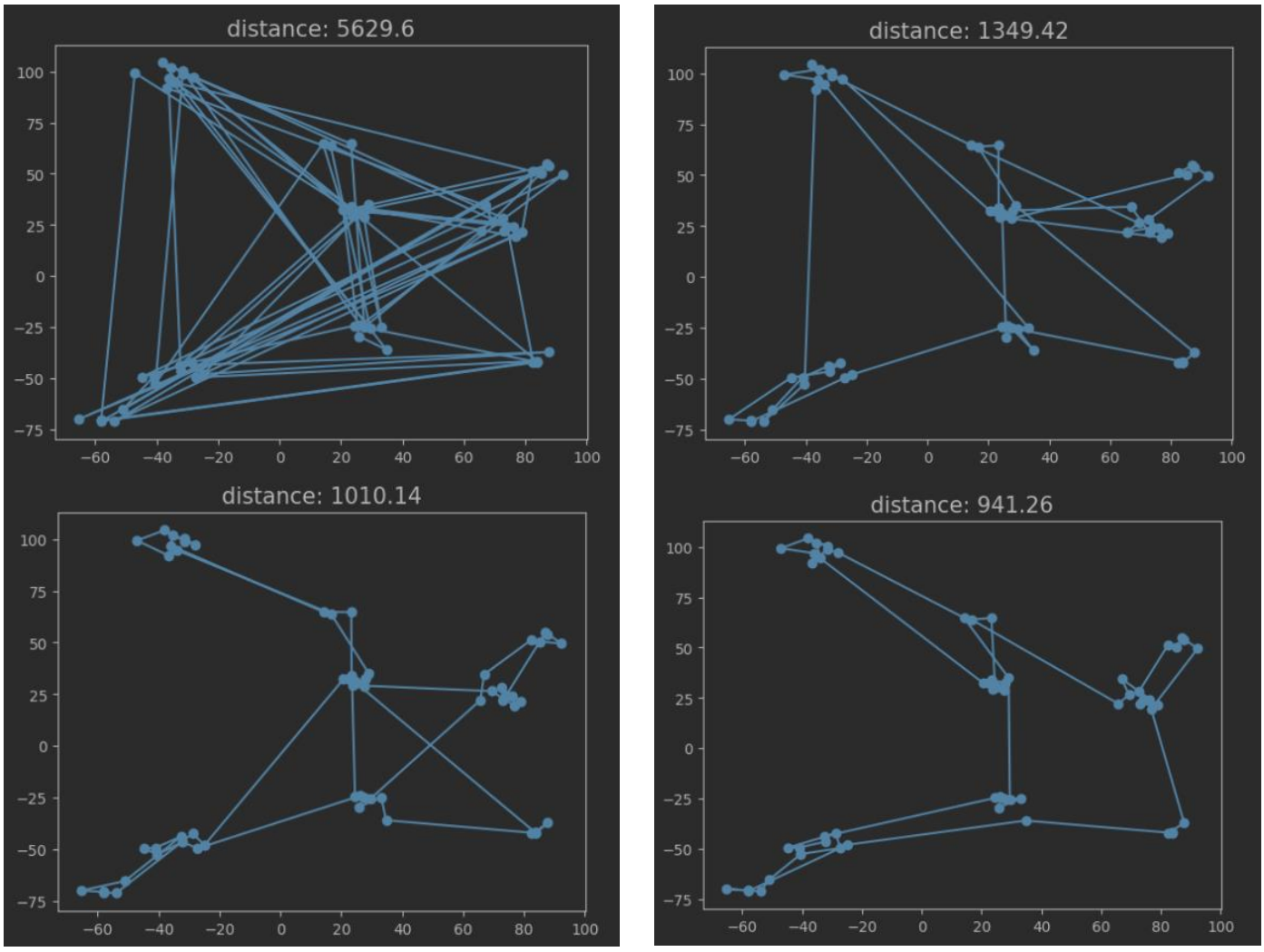


Tabela 10

b) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
sposób zamiany punktów -> zamień sąsiednie punkty

Wizualizacja :

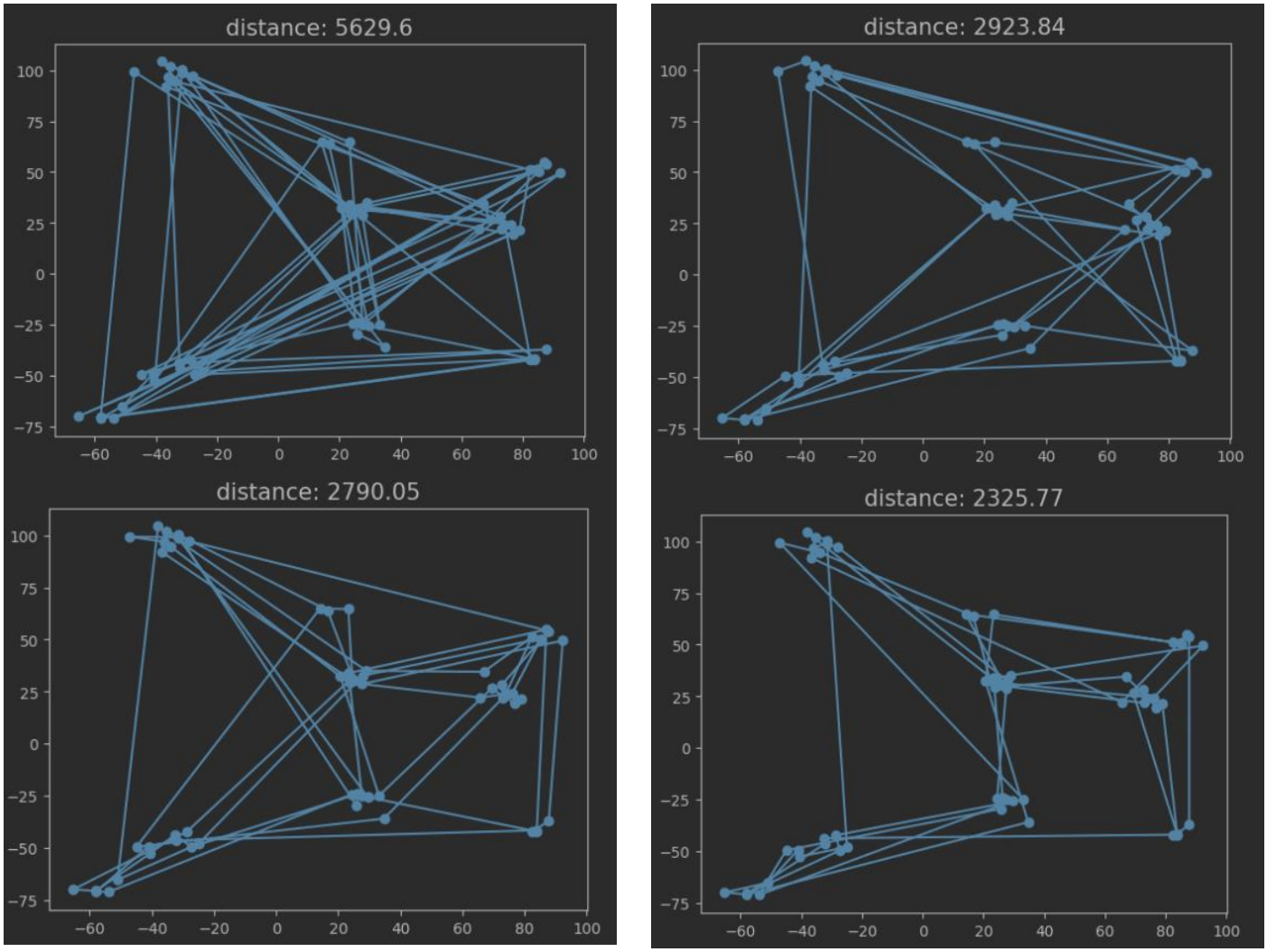


Tabela 11

c) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
sposób zamiany punktów ->
80% na zamianę losowych
20% na zamianę sąsiednich

Wizualizacja :

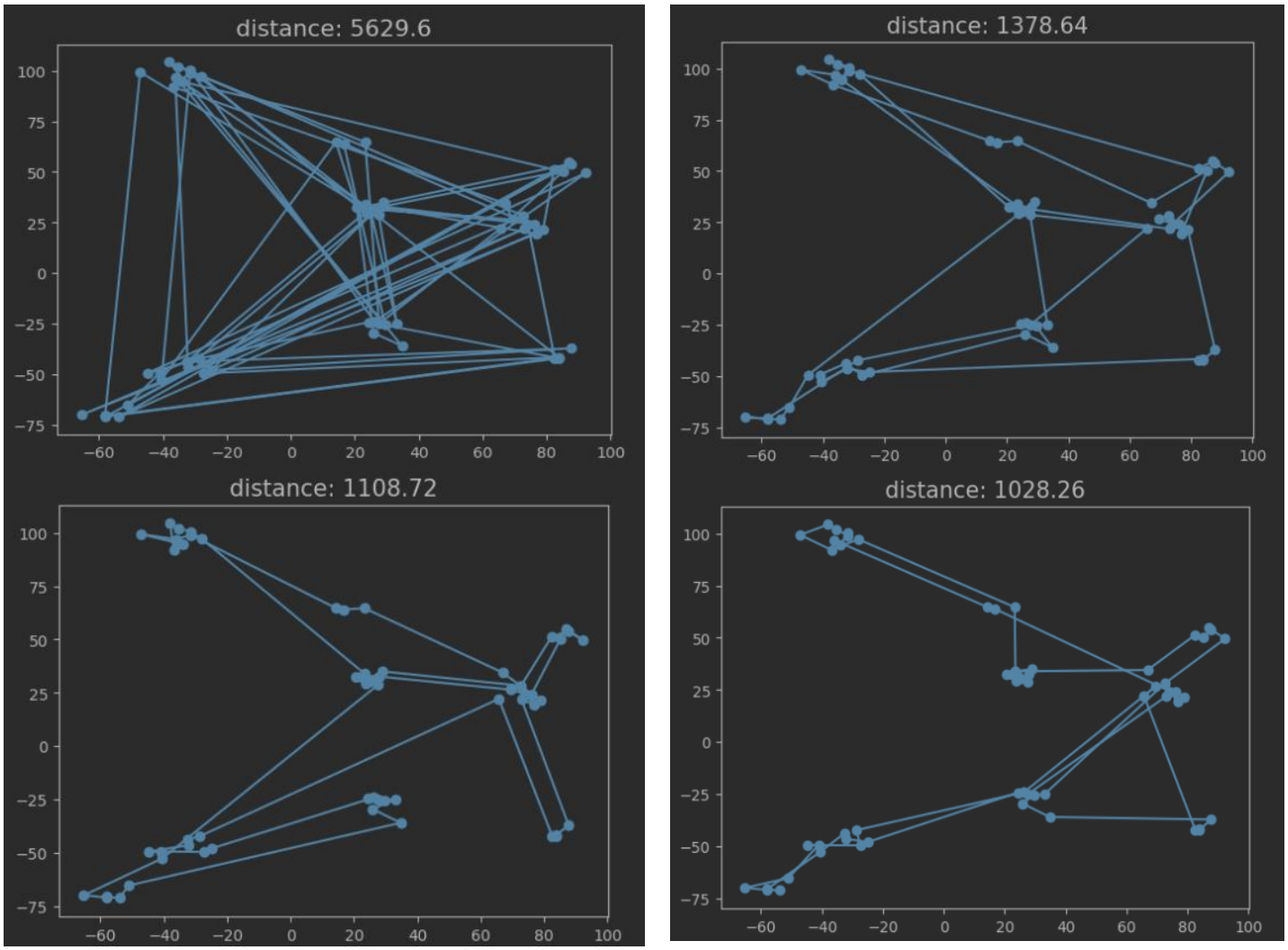
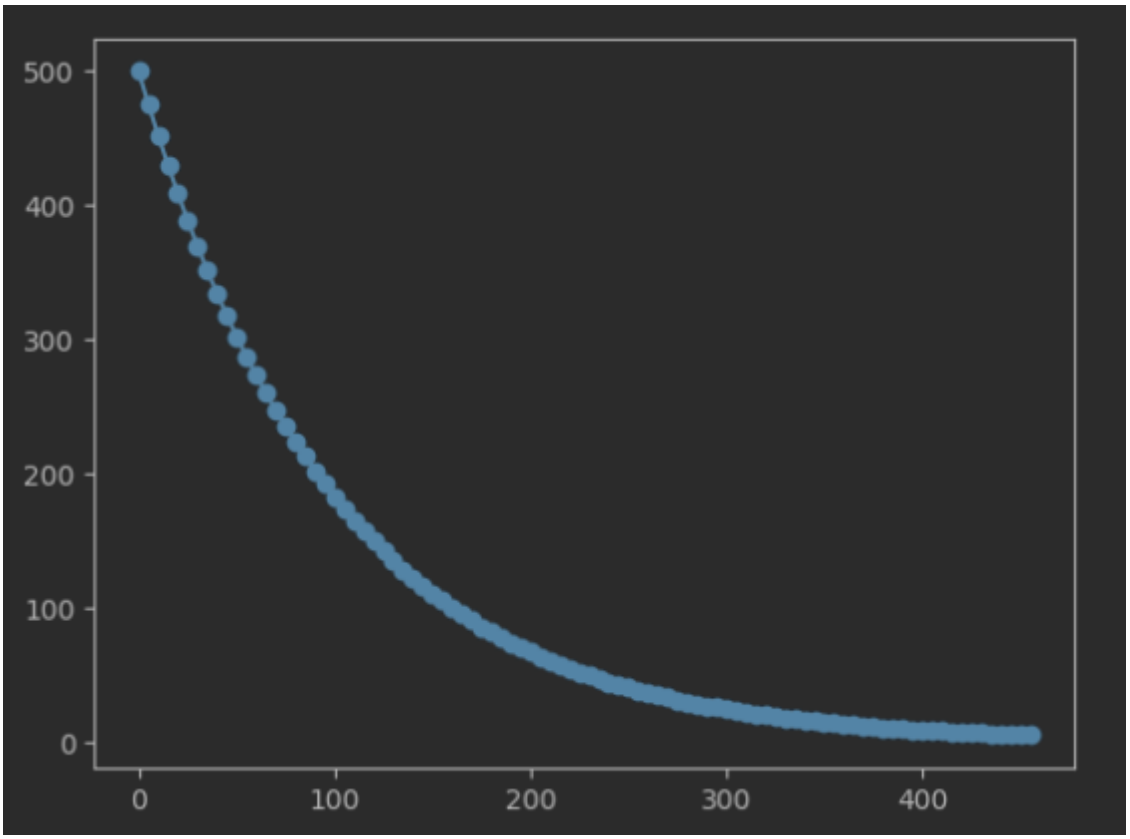


Tabela 12

1.5 Porównanie różnych funkcji temperatury dla losowo wygenerowanych 50 punktów

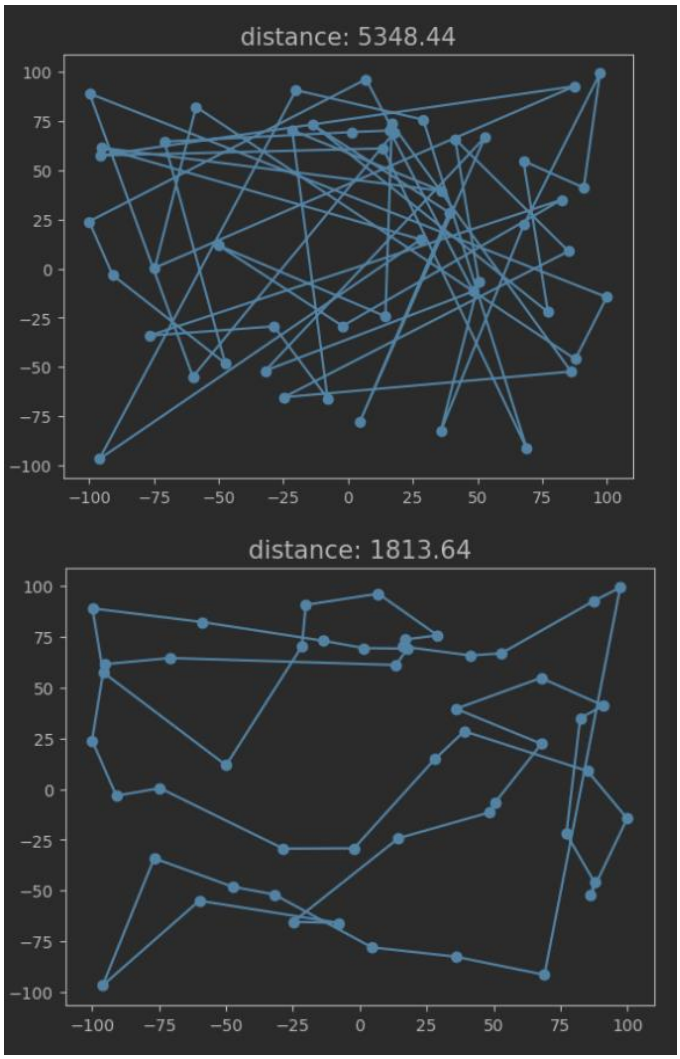
a) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.99\ t_{n-1}$
sposób zamiany punktów ->
80% na zamianę losowych
20% na zamianę sąsiednich

funkcja temperatury:



Rysunek 2

Wizualizacja :



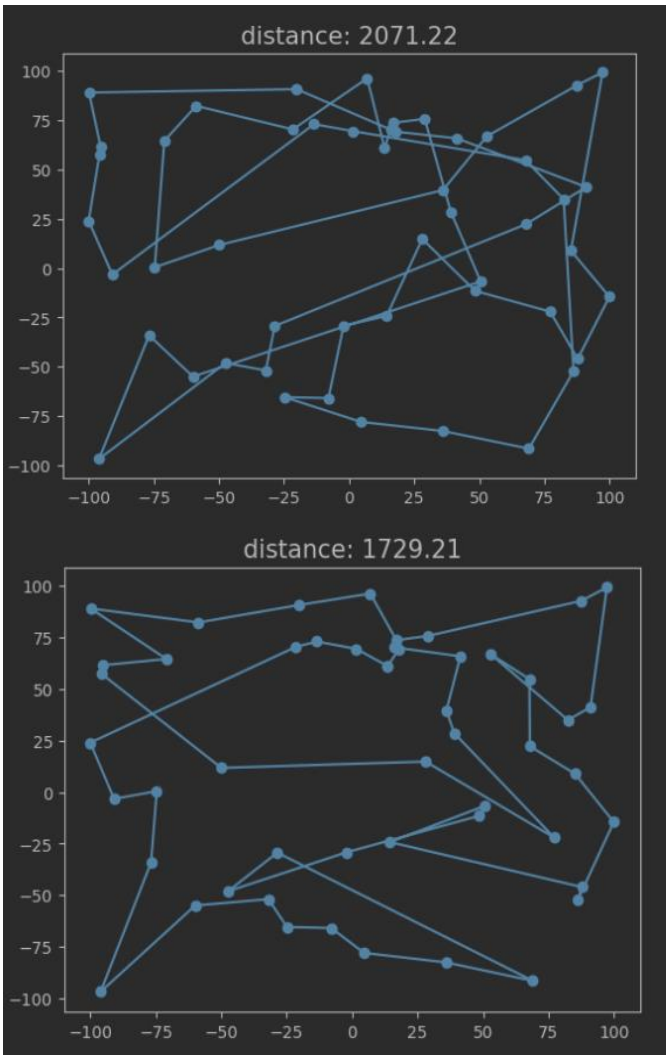
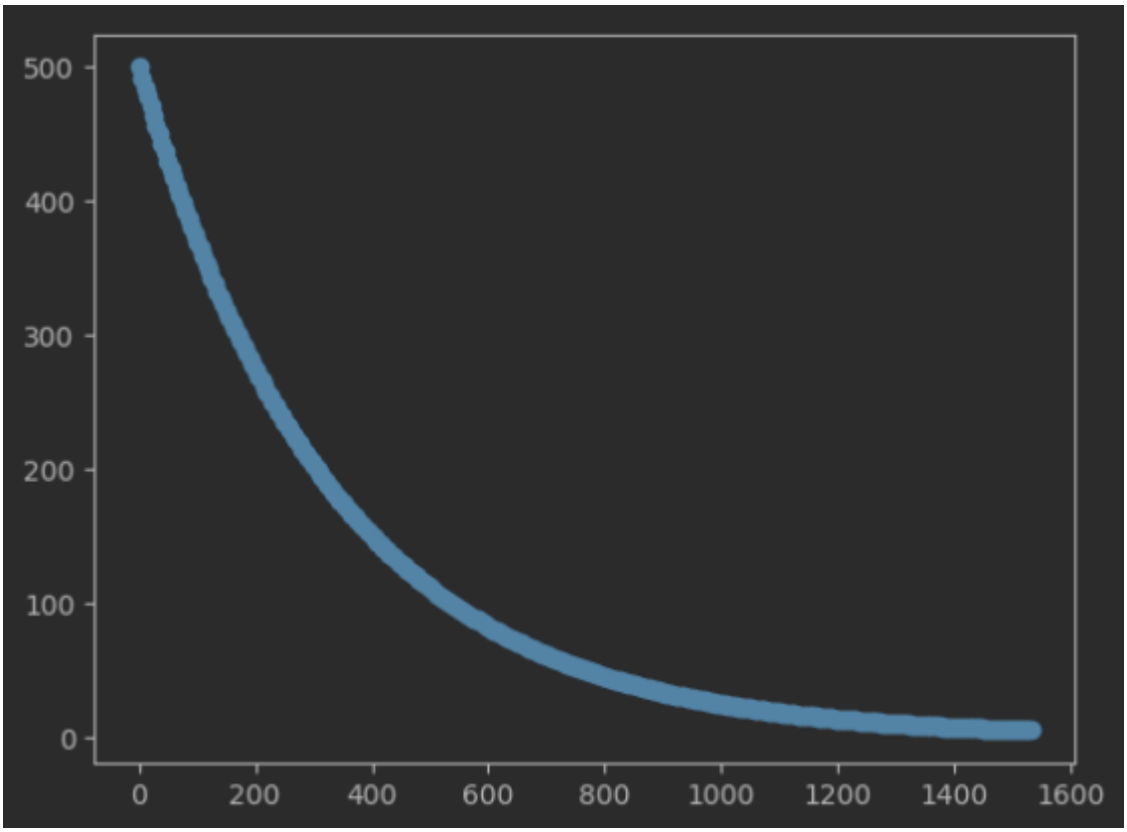


Tabela 13

b) temperatura początkowa = 500
   temperatura końcowa = 5
   funkcja spadku temperatury -> $t_n = 0.997\ t_{n-1}$
   sposób zamiany punktów ->
      80% na zamianę losowych
      20% na zamianę sąsiednich

   funkcja temperatury:



Rysunek 3

Wizualizacja :

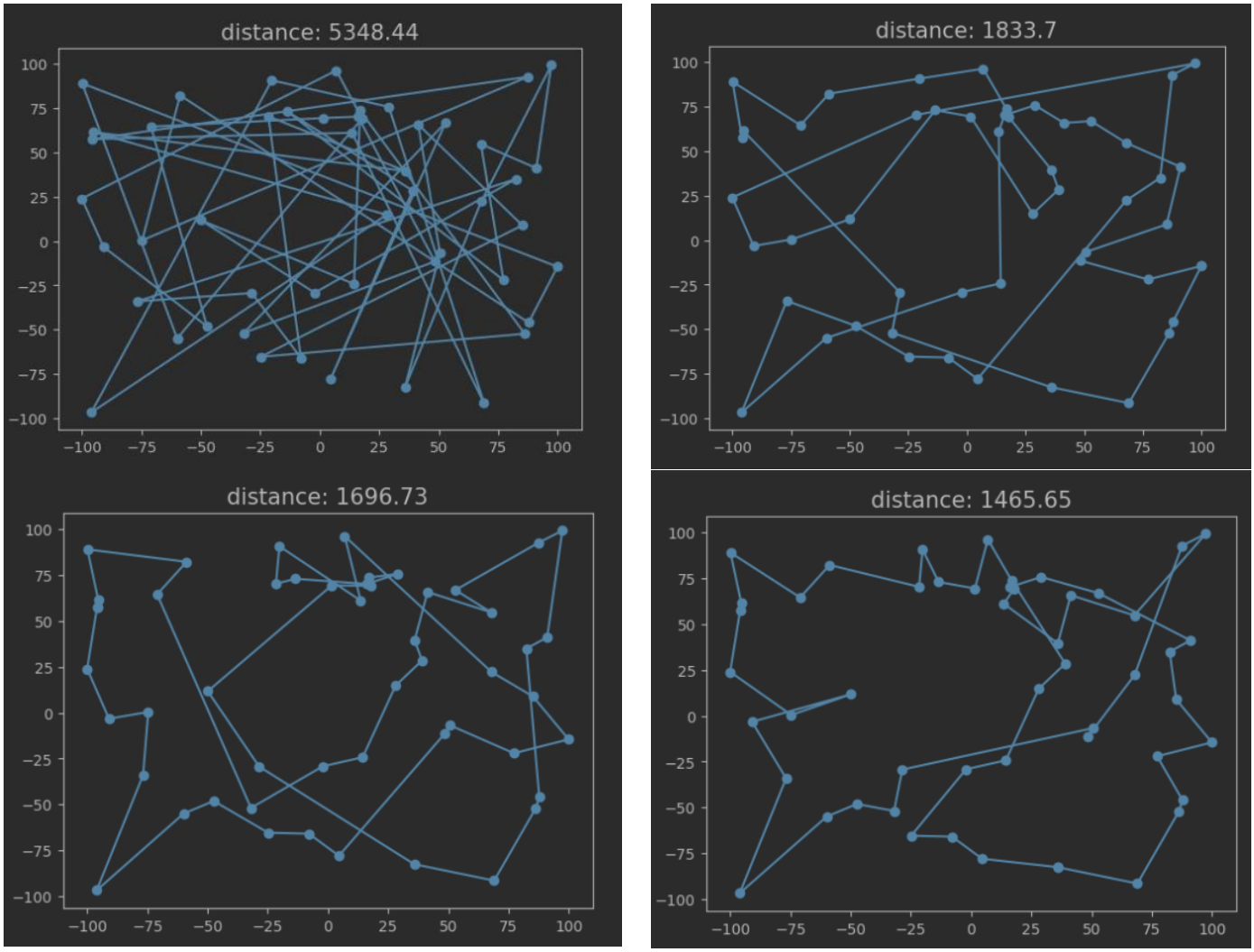
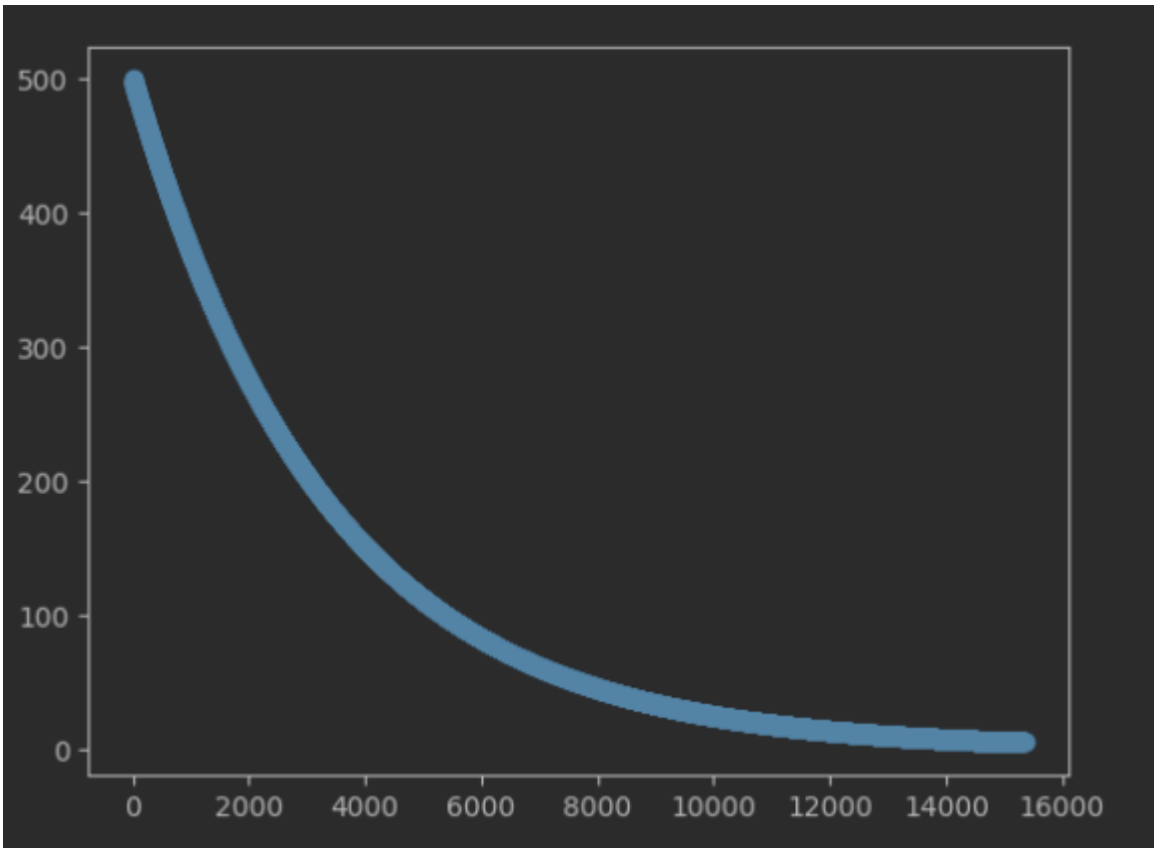


Tabela 14

c) temperatura początkowa = 500
temperatura końcowa = 5
funkcja spadku temperatury -> $t_n = 0.9997\ t_{n-1}$
sposób zamiany punktów ->
80% na zamianę losowych
20% na zamianę sąsiednich

funkcja temperatury:



Rysunek 4

Wizualizacja :

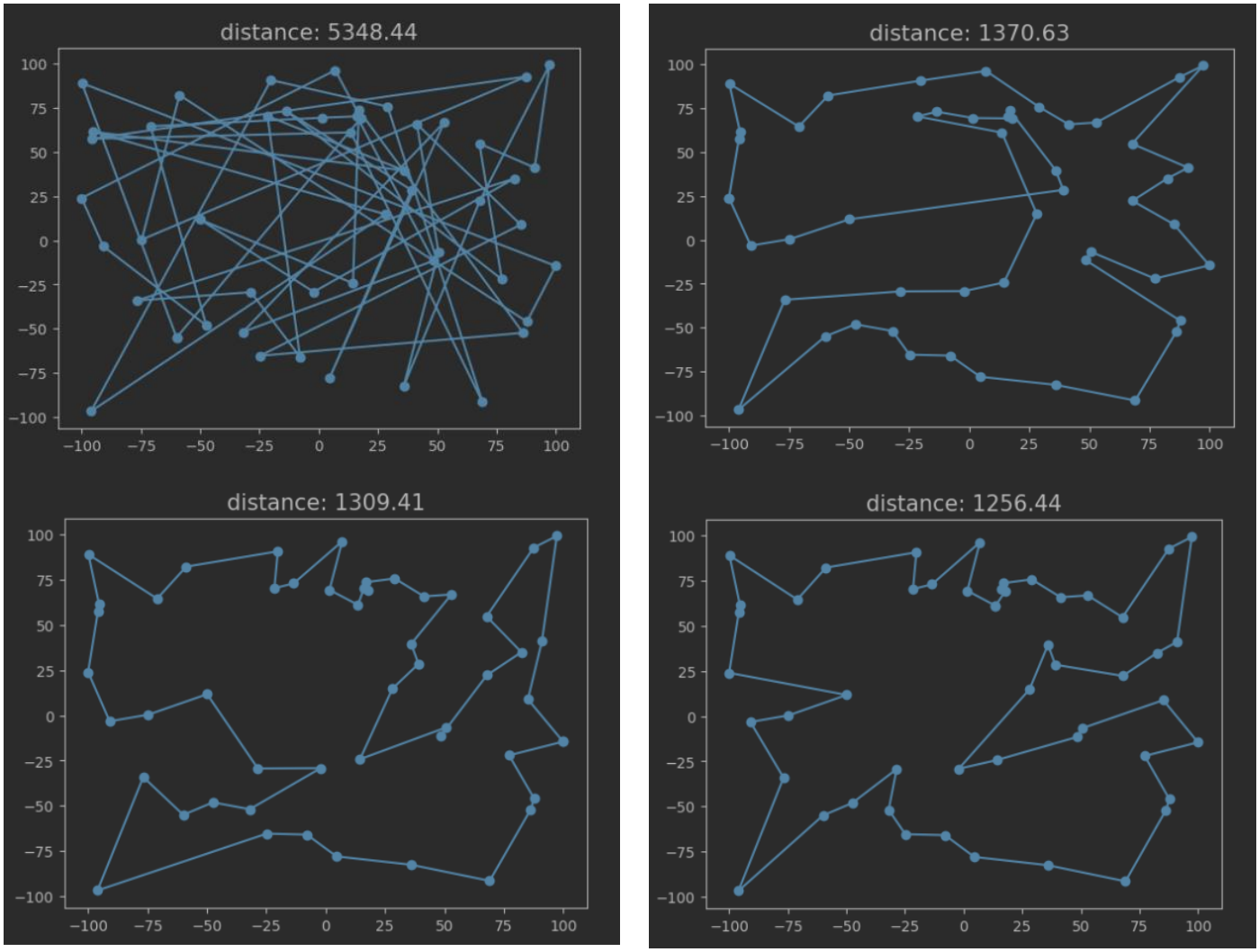
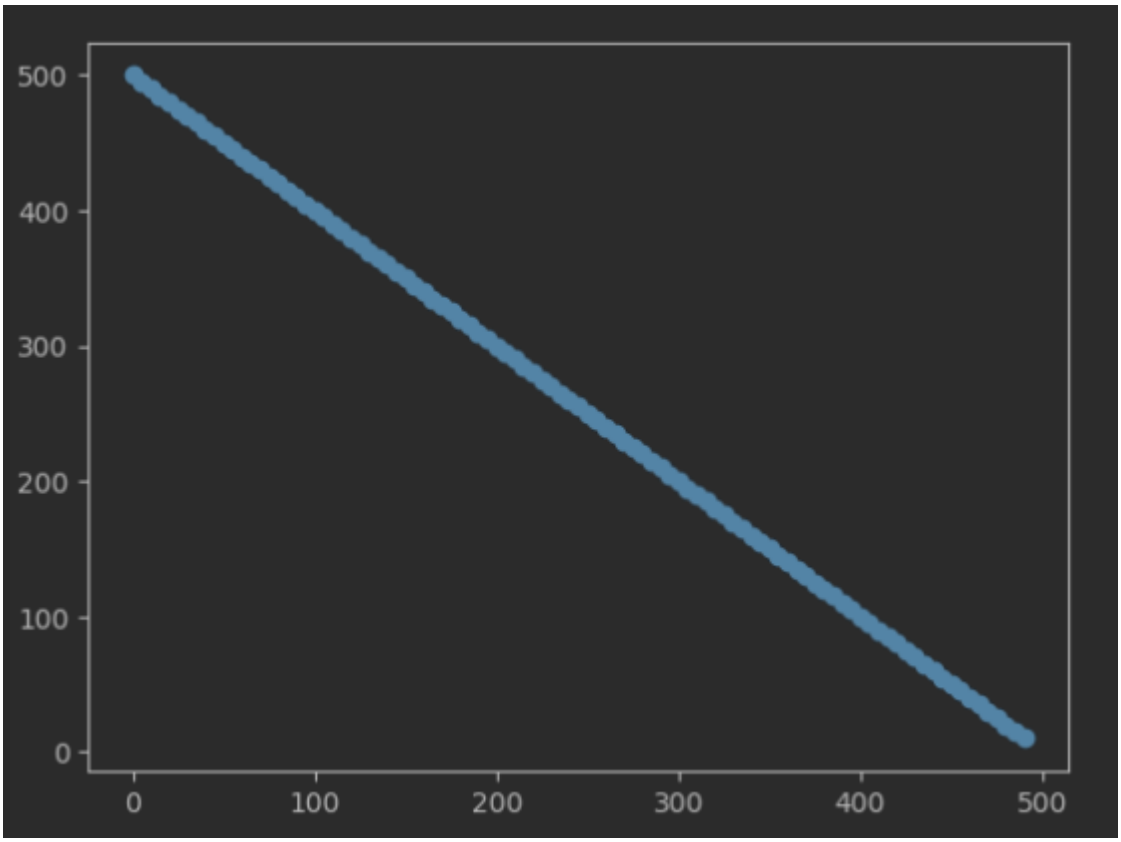


Tabela 15

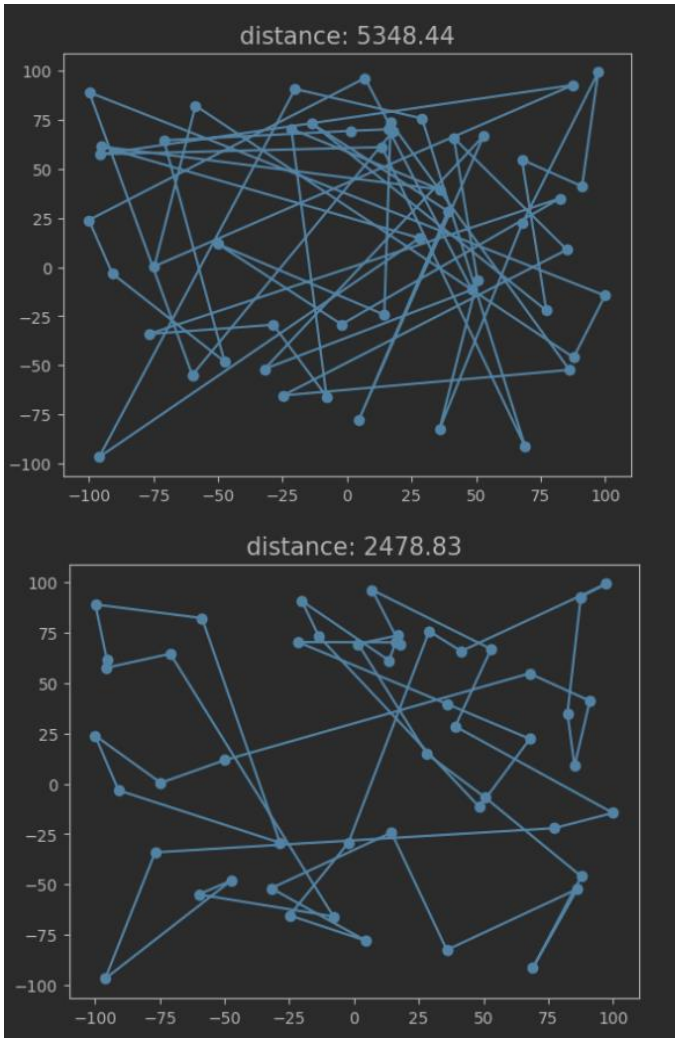
d) temperatura początkowa = 500
   temperatura końcowa = 5
   funkcja spadku temperatury -> $t_n = t_{n-1} - 1$
   sposób zamiany punktów ->
      80% na zamianę losowych
      20% na zamianę sąsiednich

   funkcja temperatury:



Rysunek 5

Wizualizacja :



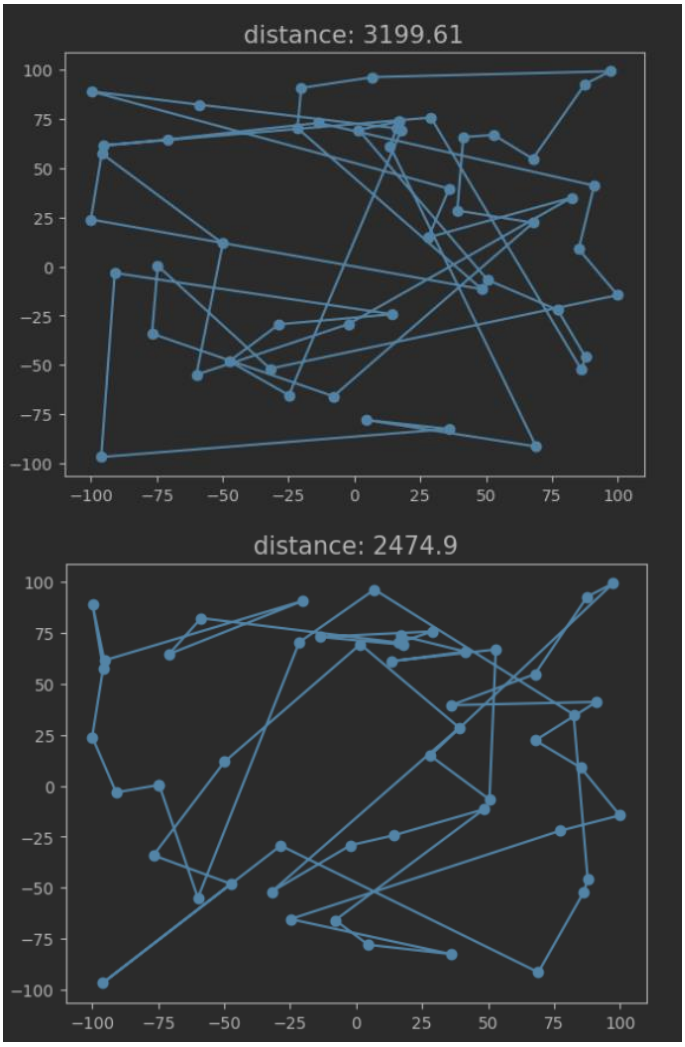


Tabela 16

Wnioski do zadania 1

Do rozwiązywania problemu podróżującego listonosza symulowane wyżarzanie jest ciekawym podejściem. Przy 100 losowych punktach niestety znajdowane rozwiązanie było dość chaotyczne, ale zwiększenie ilości iteracji i spowolnienie spadku temperatury sprawiło by, że dzięki zwiększonej liczbie iteracji znalezione rozwiązanie było by bliższe optymalnemu. Przy każdym z przeprowadzonych testów okazywało się, że zamienianie ze sobą dwóch losowych punktów było lepsze, niż zamienianie dwóch sąsiednich. Natomiast przy połączeniu tych dwóch sposobów zamian w praktycznie każdym z przeprowadzonych testów uzyskano jeszcze lepszy rezultat końcowy, bez dodatkowego zwiększania ilości iteracji.

Zadanie 2

Zacząłem od wygenerowania obrazów bitowych 200x200 w których było tyle samo pikseli czarnych i białych. Później w zależności od przyjętych funkcji kosztu powstawały różne obrazy.

1. Koszt rośnie o 1 za każdego sąsiada każdego piksela, który jest inny niż on sam (liczymy tylko 8 najbliższych sąsiadów)

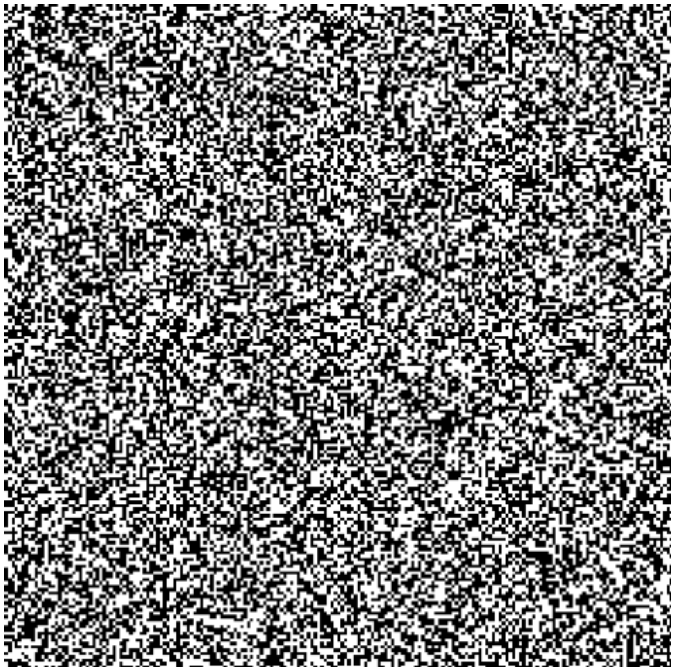
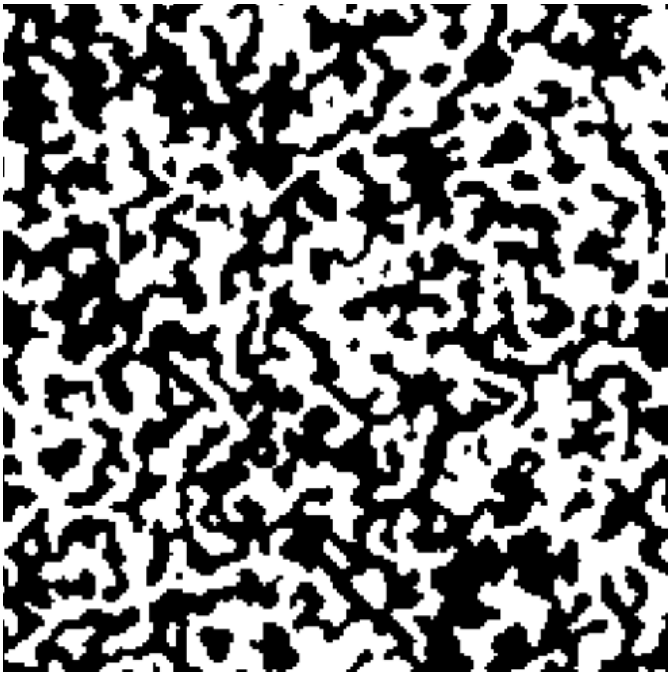
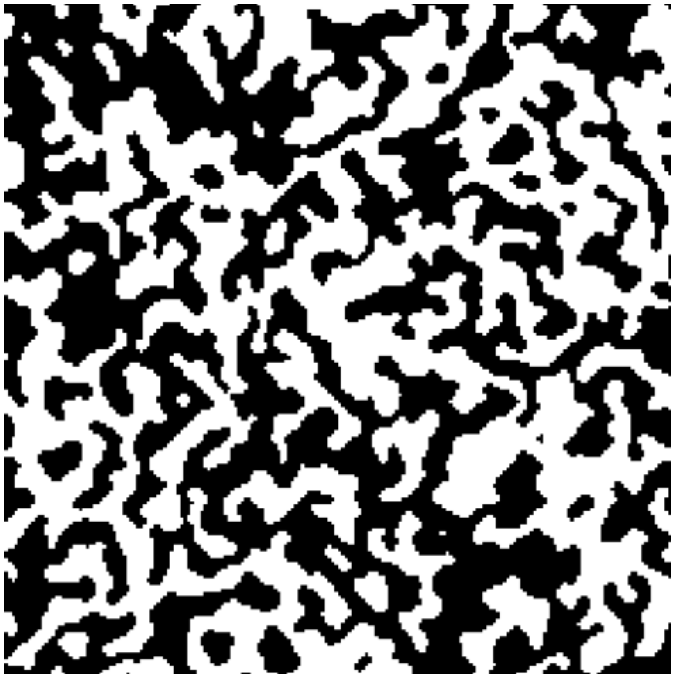
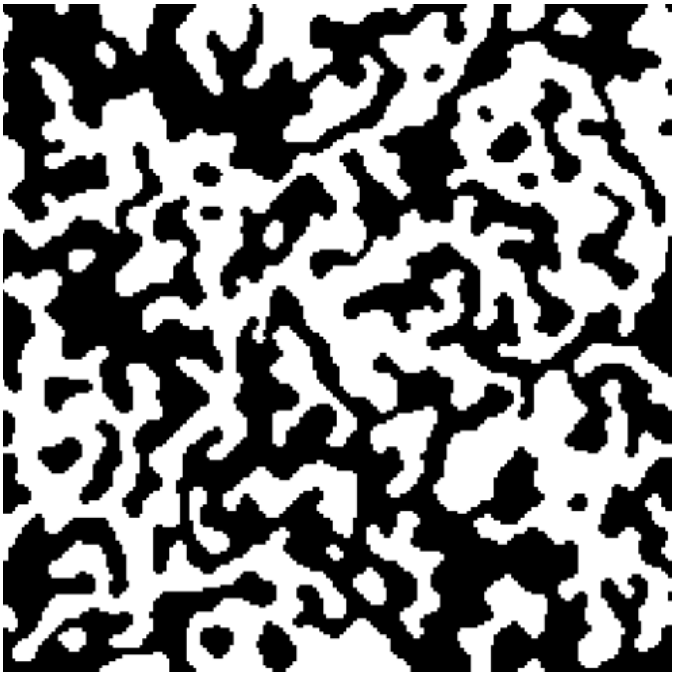





Tabela 17

2. Koszt rośnie o 1 za każdego sąsiada każdego piksela, który jest inny niż on sam (liczymy tylko 4 najbliższych sąsiadów)

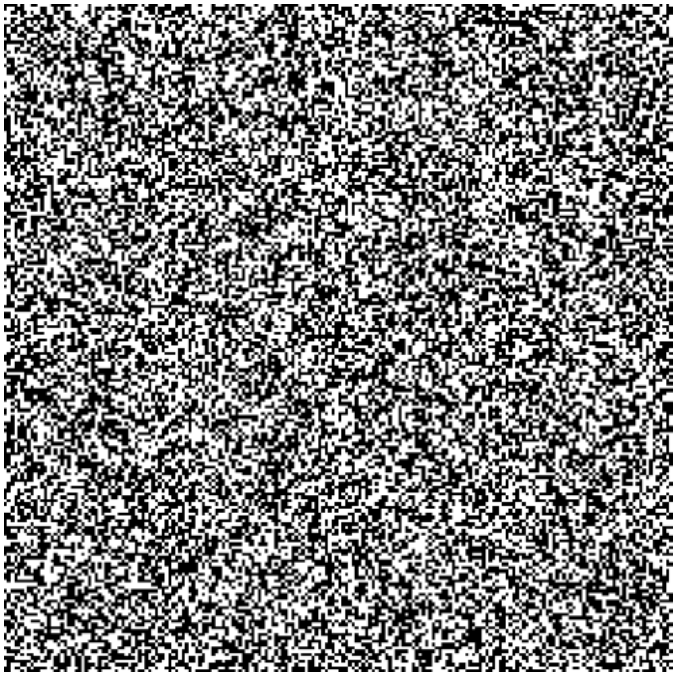
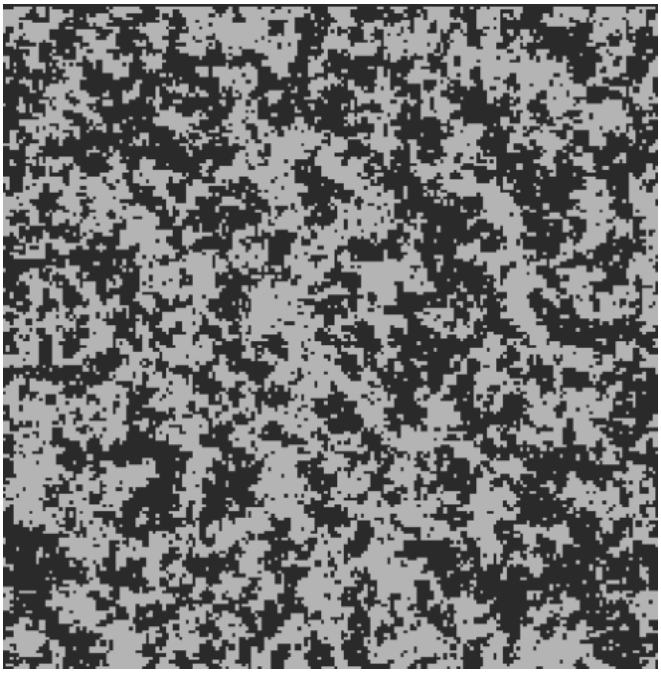
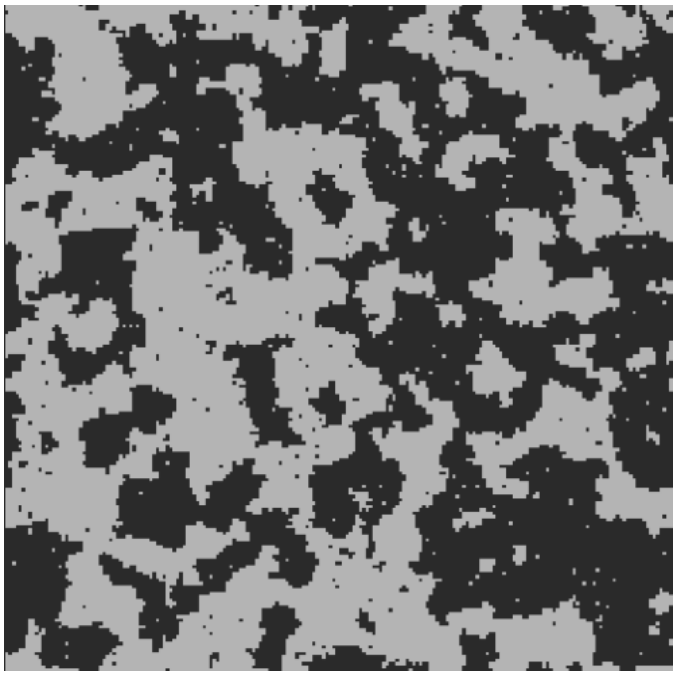
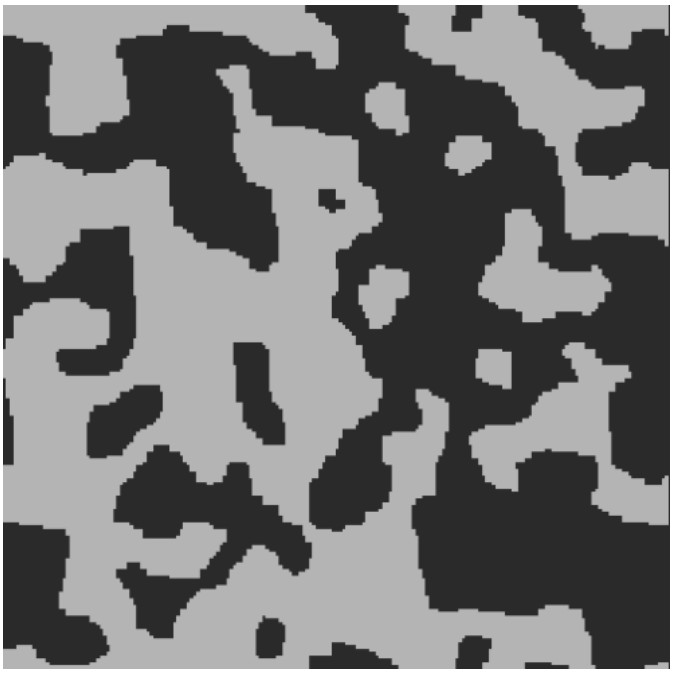





Tabela 18

3. Koszt rośnie o 1 za każdego sąsiada każdego piksela, który jest tak sam jak on sam (liczymy tylko 4 najbliższych sąsiadów) (powinniśmy zobaczyć „szachownicę”)

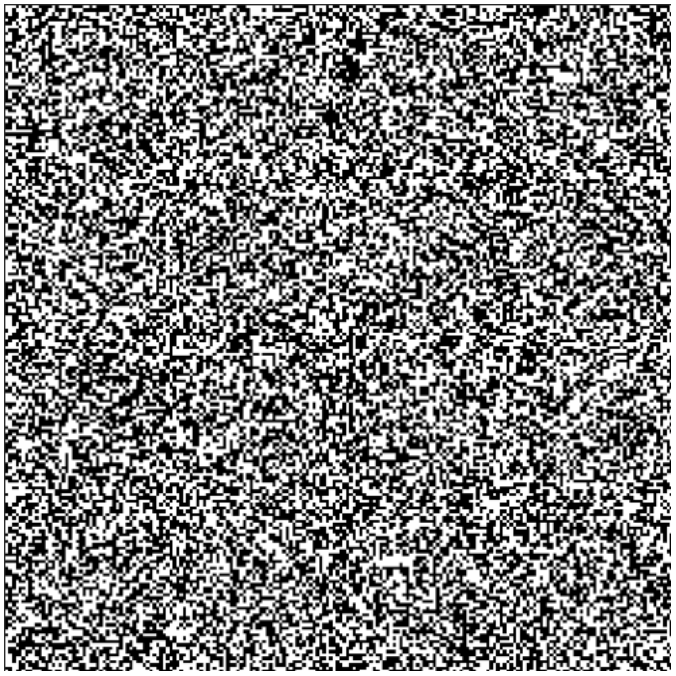
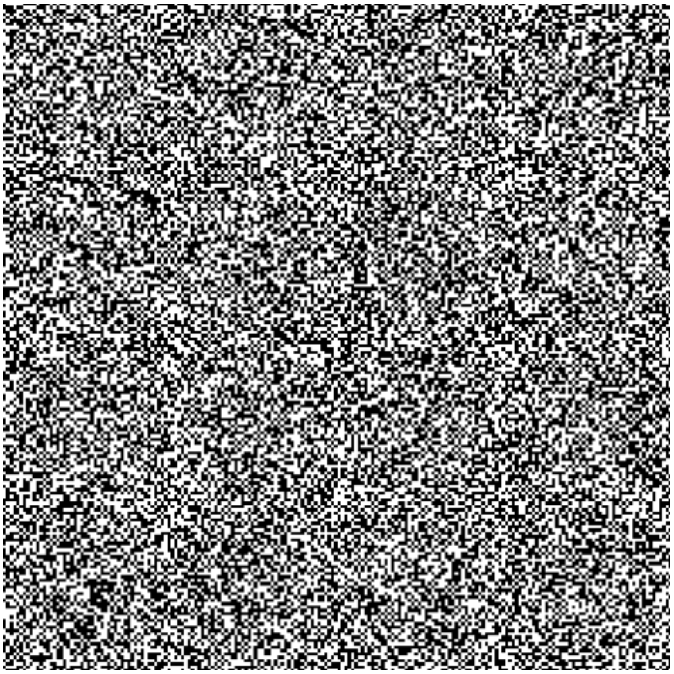
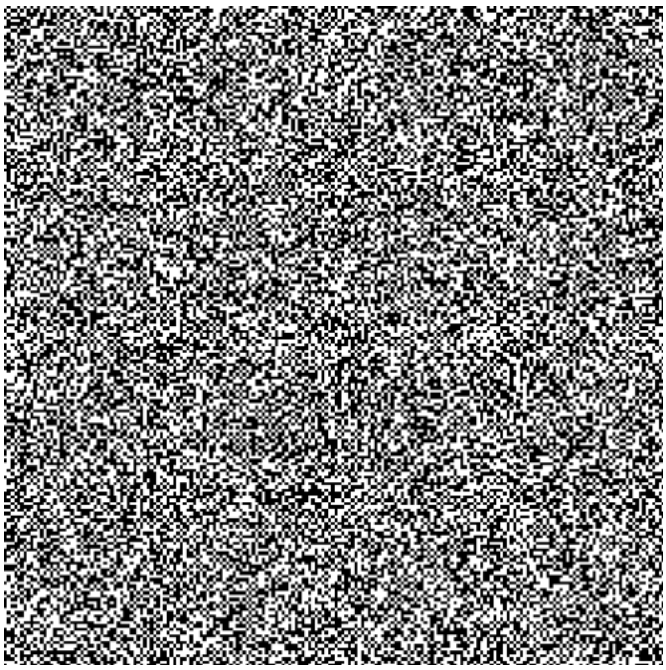
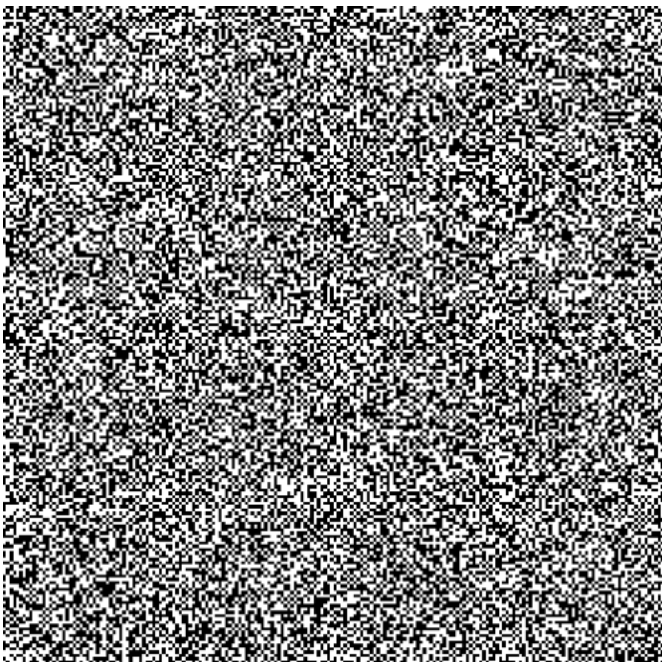





Tabela 19

4. Koszt rośnie o 1 za każdego sąsiada każdego piksela, który jest tak sam jak on sam z góry lub z dołu, lub różny z lewej lub prawej

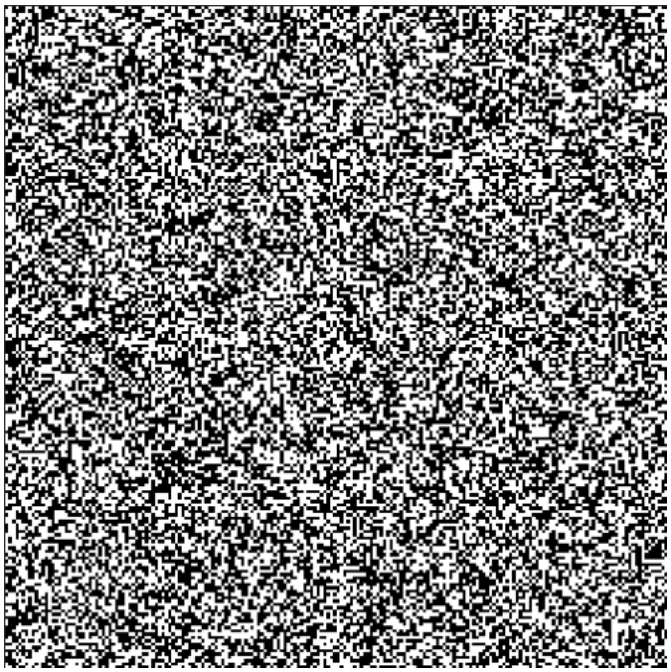
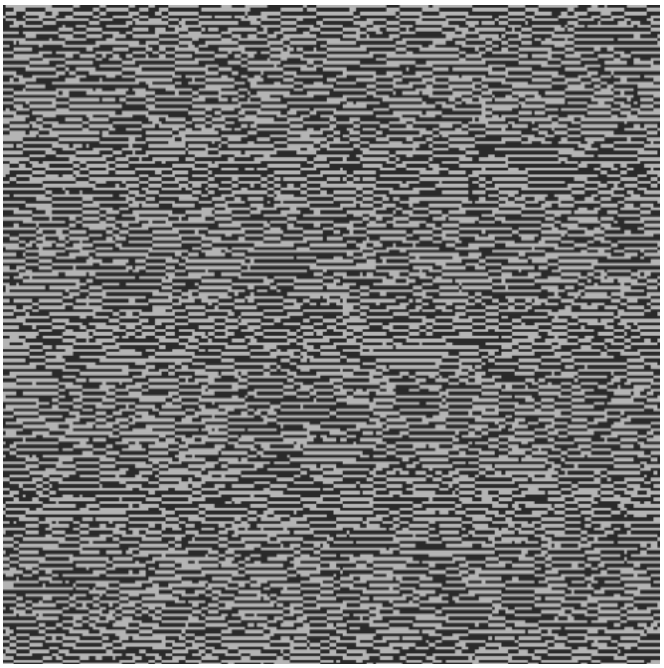
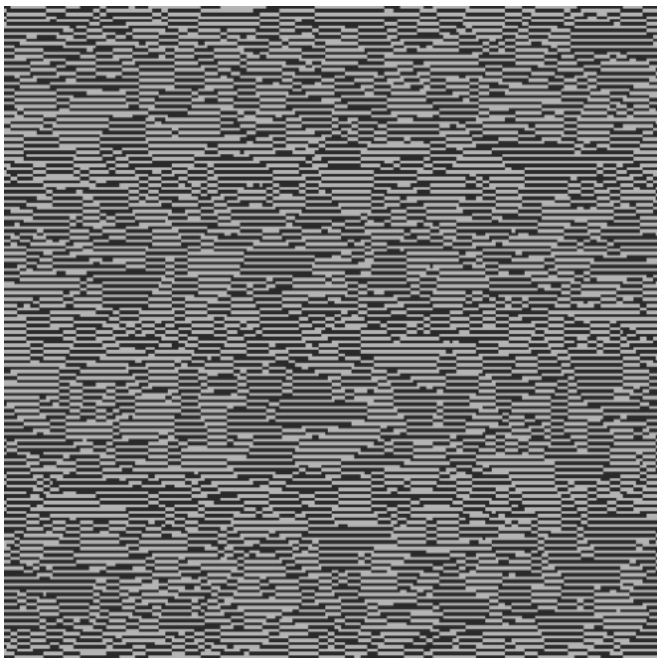
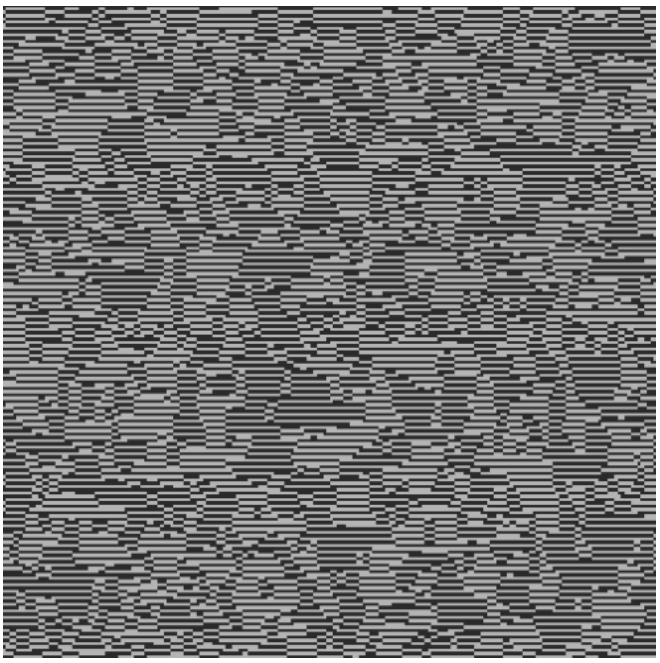
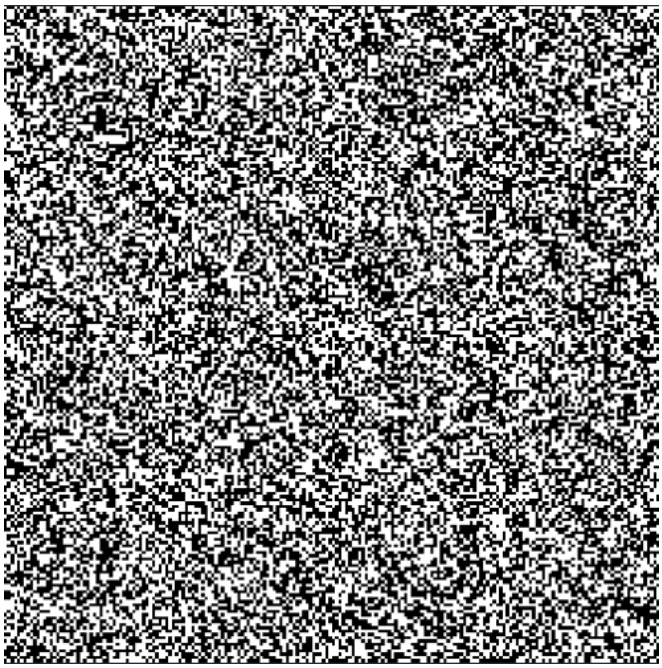
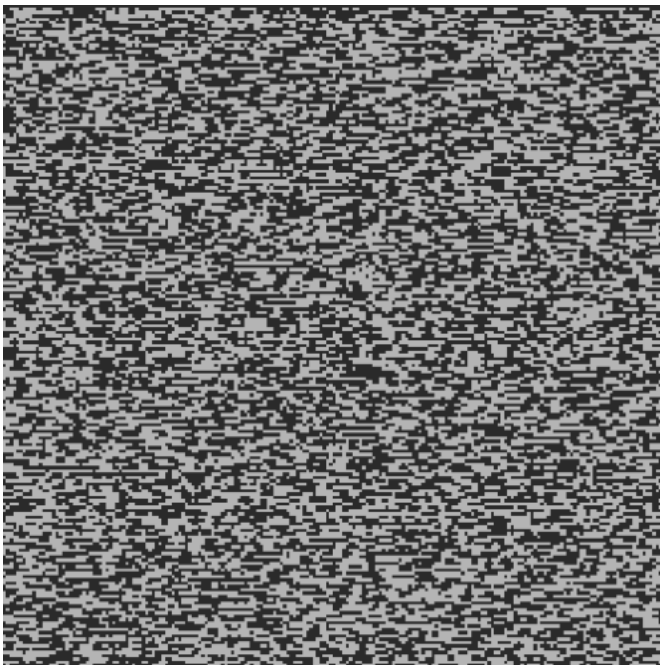
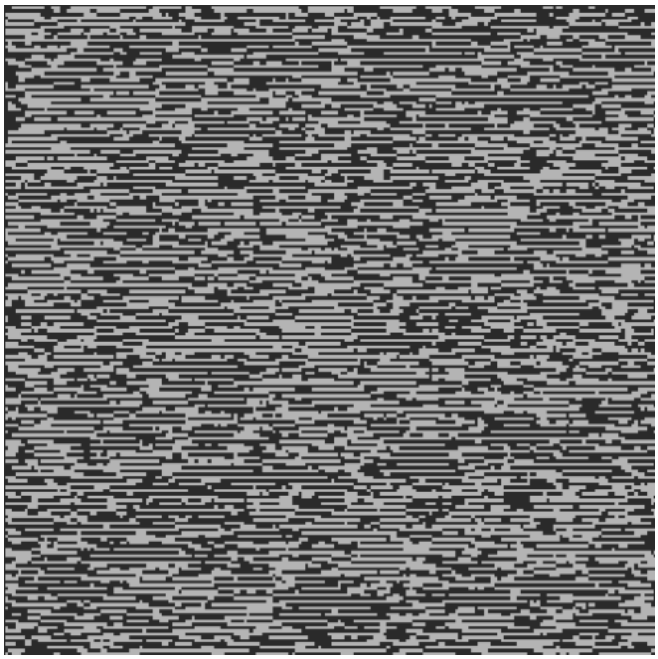





Tabela 20

5. Koszt rośnie o 1 za każdego sąsiada każdego piksela, który jest tak sam jak on sam z góry, lub różny z lewej, z prawej lub z dołu







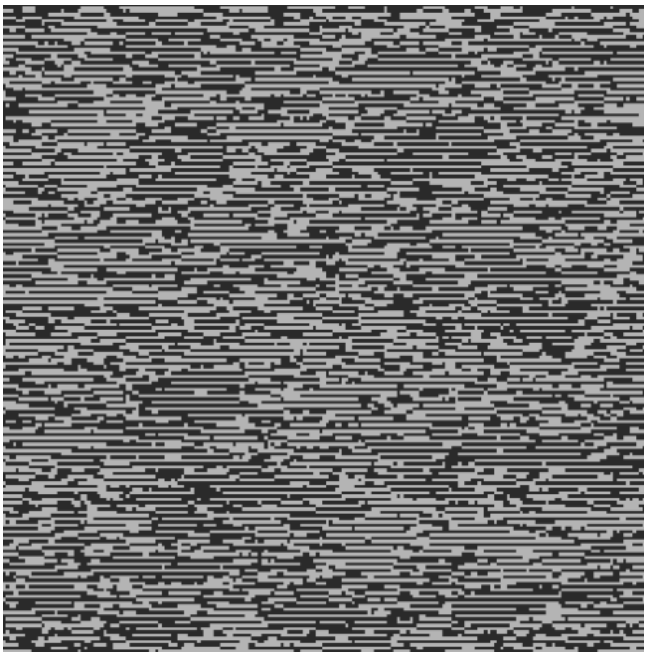


Tabela 21

Wnioski do zadania 2

Aby powstały fajne obrazy trzeba najpierw wymyślić ciekawą funkcję kosztu. Jeśli funkcja kosztu nie ma sensu, to obraz nigdy nie dojdzie do sensownego stanu, a wszystkie zamiany będą po prostu go losowo zmieniać. W tym przypadku powinno się zaczynać od niższej temperatury początkowej, ponieważ obraz sam w sobie jest losowy, więc losowe zamiany nie mają dużego znaczenia i tylko spowalniają powstawanie ciekawych wzorów. Dodatkowo aby obrazy miały szansę znacząco obniżyć swój koszt, funkcja spadku temperatury powinna powodować bardzo powolne spadki. W moich obrazach wynosiła ona $t_n = 0.9999995\ t_{n-1}$, co dla początkowej temperatury wynoszącej 8, i końcowej 0.1 (dodatkowo dla każdego poziomu temperatury było wykonywanych 10 iteracji) dawało łącznie 87 640 520 iteracji (tyle iteracji były w obrazie z podpunktu 2, w innych ta liczba była inna z powodu innych stanów początkowych, lub przerwaniu liczenia kolejnych iteracji z powodu braku zadowalających zmian obrazu).

Wnioski do symulowanego wyżarzania

Na początku chciałem stworzyć swoją własną mini bibliotekę, którą mógłbym uniwersalnie wykorzystywać do różnych zadań z tego laboratorium (oraz w przyszłości, jeśli była by taka potrzeba). Niestety po długich godzinach zrozumiałem, że symulowane wyżarzanie najlepiej jest pisać od zera dla praktycznie każdego napotkanego problemu, ponieważ mogą być potrzebne małe zmiany, które były by bardzo ciężkie (lub wręcz niemożliwe) do zaimplementowania z bibliotece ogólnej, a które bardzo pomogły by nam przy rozwiązaniu naszego problemu. Na samym początku rozwiązywania warto poeksperymentować i zastanowić się jakie stany początkowe temperatury, oraz jaka funkcja spadku temperatury najbardziej pomogą nam rozwiązać problem, ponieważ uruchomienie programu ze złymi parametrami może spowodować utratę kilkudziesięciu minut (lub kilku godzin dla większych problemów) pracy komputera bez osiągnięcia jakichkolwiek efektów.

Symulowane wyżarzanie to ciekawa i przydatna heurystyka. Dobrze zastosowana z pewnością jest w stanie przyśpieszyć proces znalezienia zadowalającego nas rozwiązania.