SPRAWOZDANIE III

METODY OBLICZENIOWE W NAUCE I TECHNICE

SYMULOWANE WYŻARZANIE

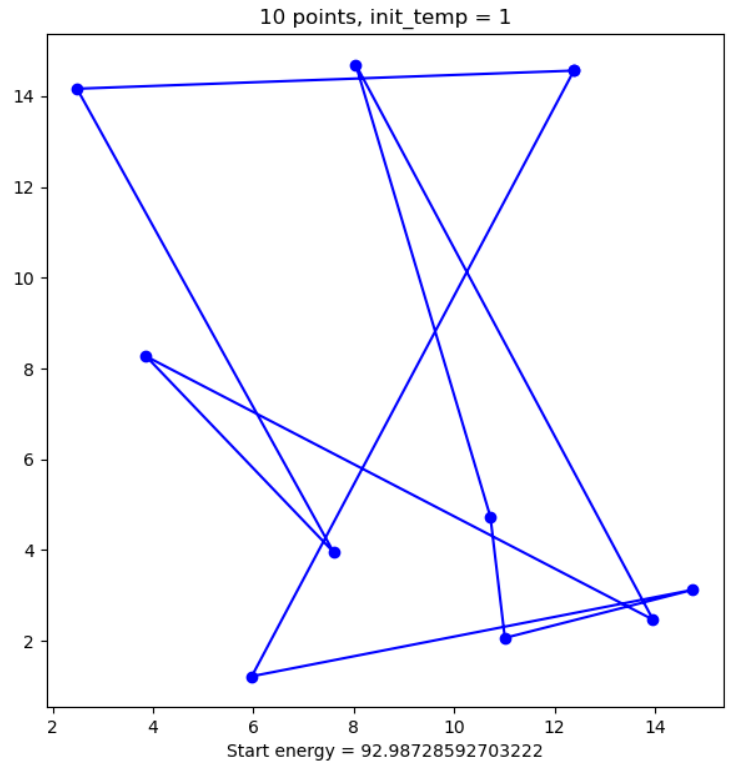
DAWID BIAŁKA

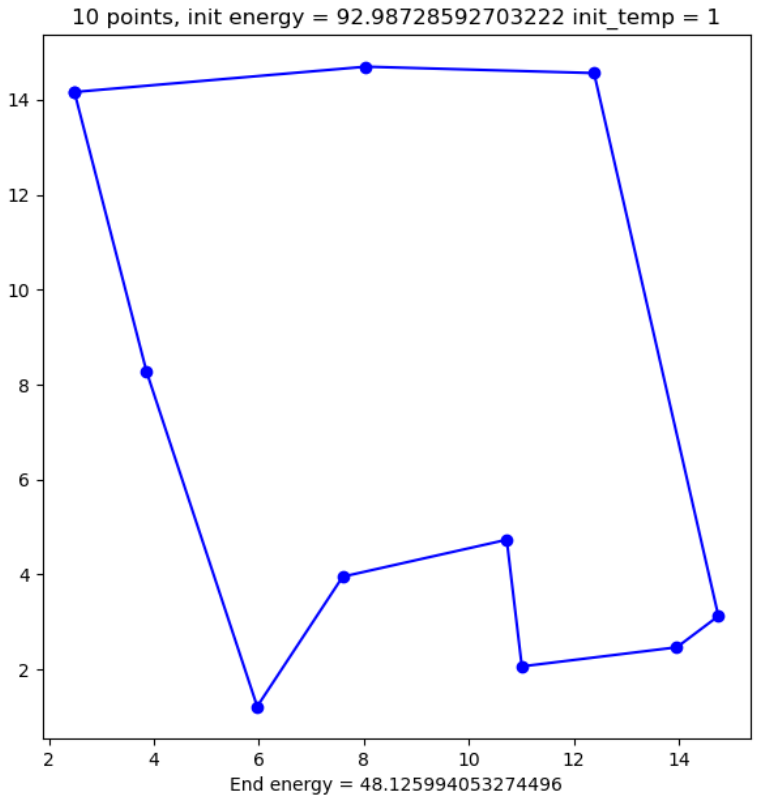
2019/2020

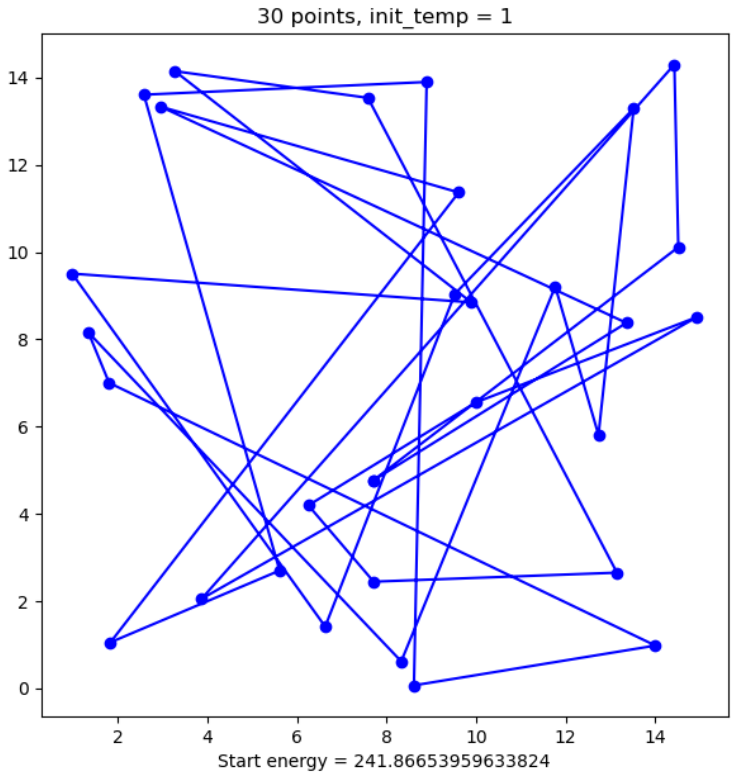
Zadanie 1 TSP

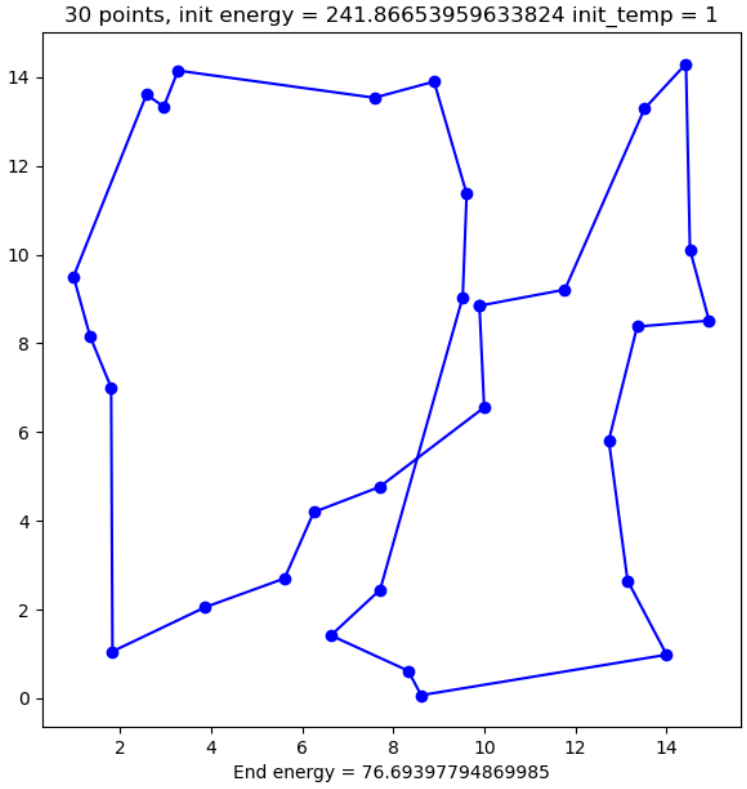
W zadaniu 1 wykorzystujemy algorytm wyżarzania do przybliżonego rozwiązania problemu komiwojażera. W trakcie działania funkcji zapamiętujemy najlepsze rozwiązanie i je później zwracamy.

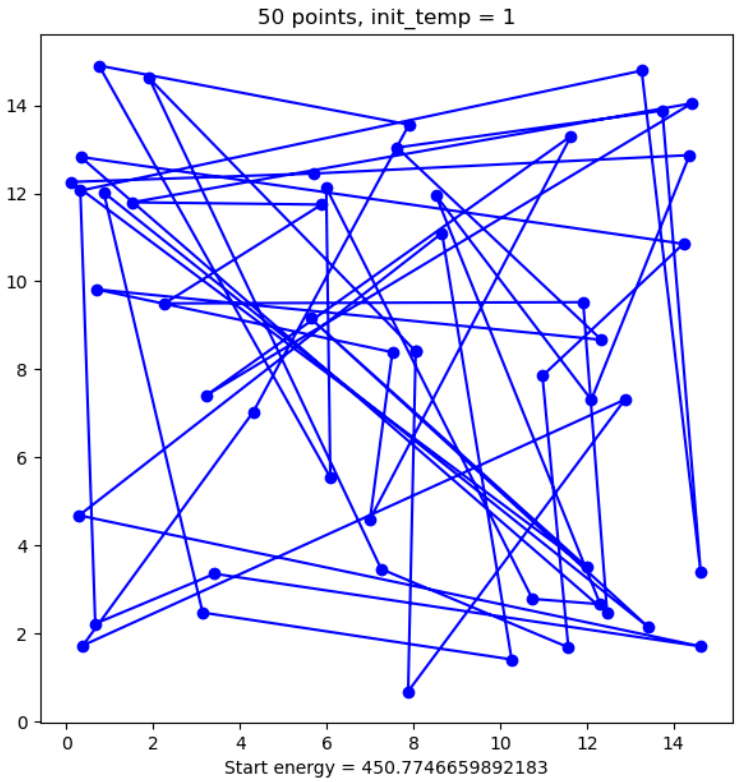
a) Ustawiamy generowanie z rozkładem jednostajnym, temperaturę początkową na 1 , generowanie sąsiadów arbitrary, 10000 iteracji i funkcję temperatury initial_temp / (iteration + 1). Generujemy wynik dla 10, 30 i 50 punktów.

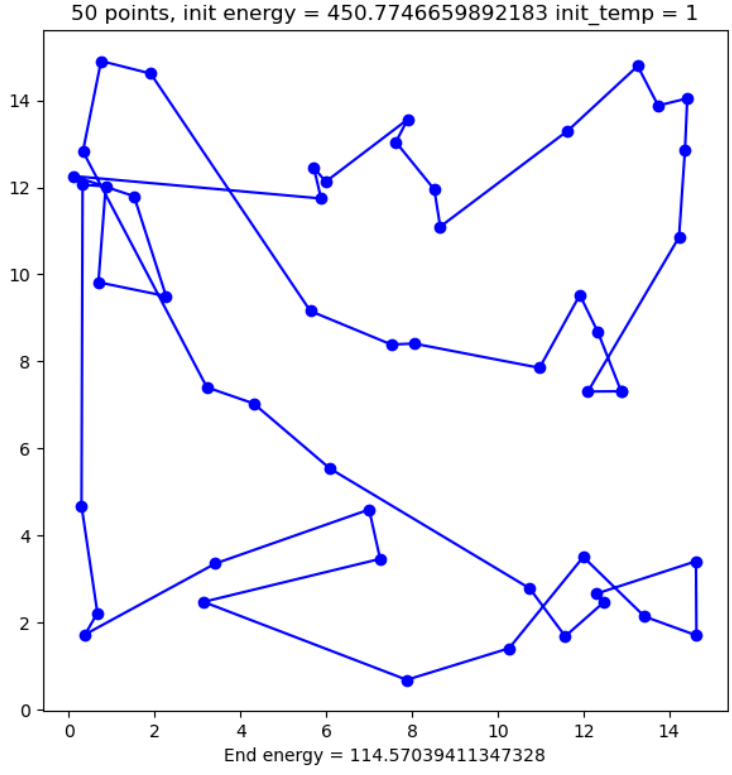

10 points, init_temp = 1
14
12
10
8
6
4
2
2
4
6
8
10
12
14
Start energy = 92.98728592703222


10 points, init energy = 92.98728592703222 init_temp = 1
14
12
10
8
6
4
2
2
4
6
8
10
12
14
End energy = 48.125994053274496

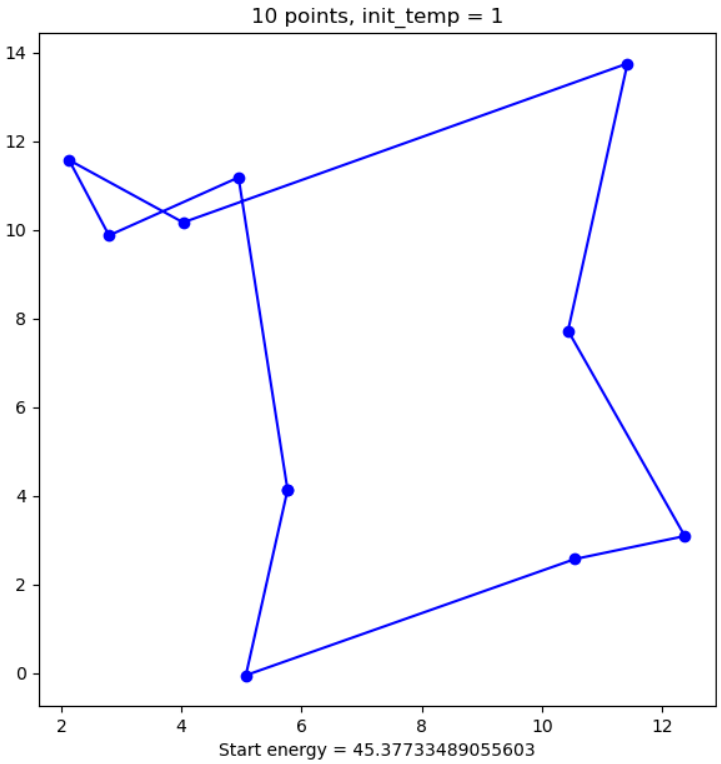

30 points, init_temp = 1
0
2
4
6
8
10
12
14
Start energy = 241.86653959633824

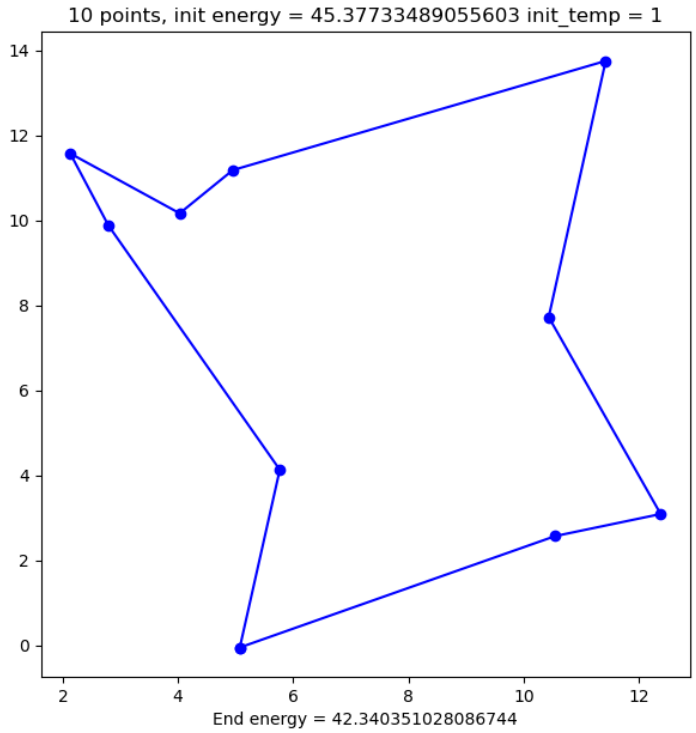

30 points, init energy = 241.86653959633824 init_temp = 1
0
2
4
6
8
10
12
14
End energy = 76.69397794869985

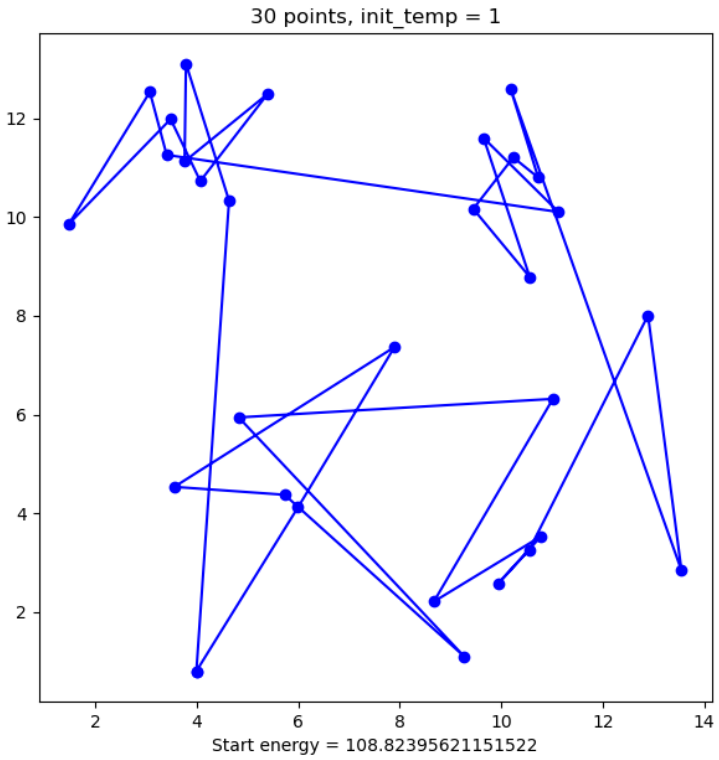

50 points, init_temp = 1
0
2
4
6
8
10
12
14
Start energy = 450.7746659892183

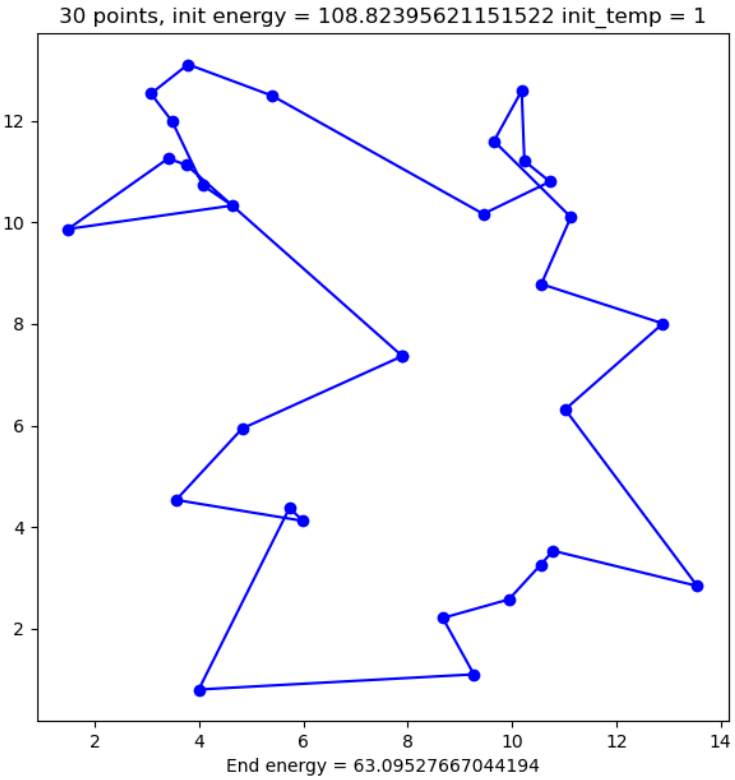

50 points, init energy = 450.7746659892183 init_temp = 1
0
2
4
6
8
10
12
14
End energy = 114.57039411347328

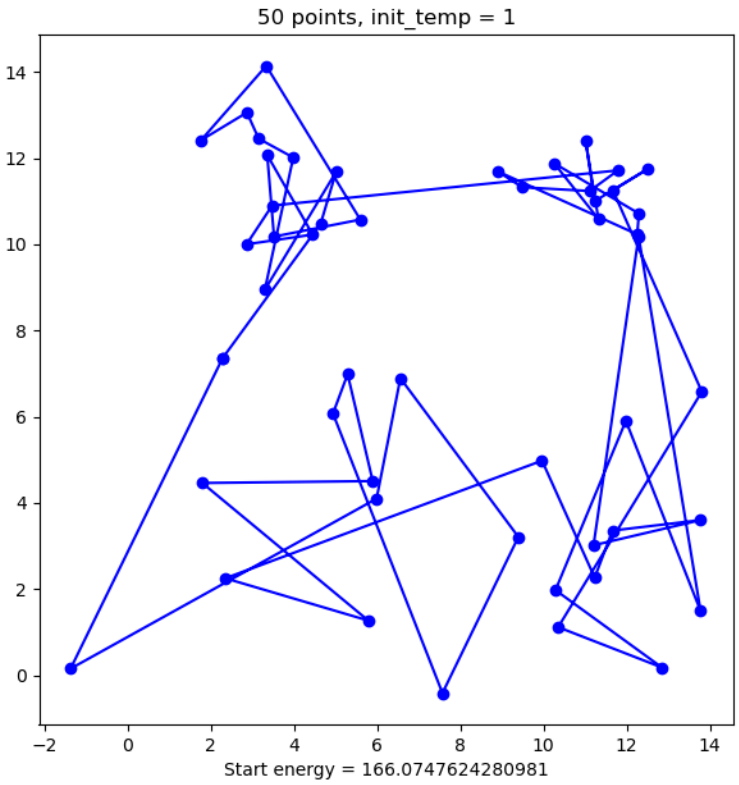
Dla generowania z rozkładem jednostajnym:

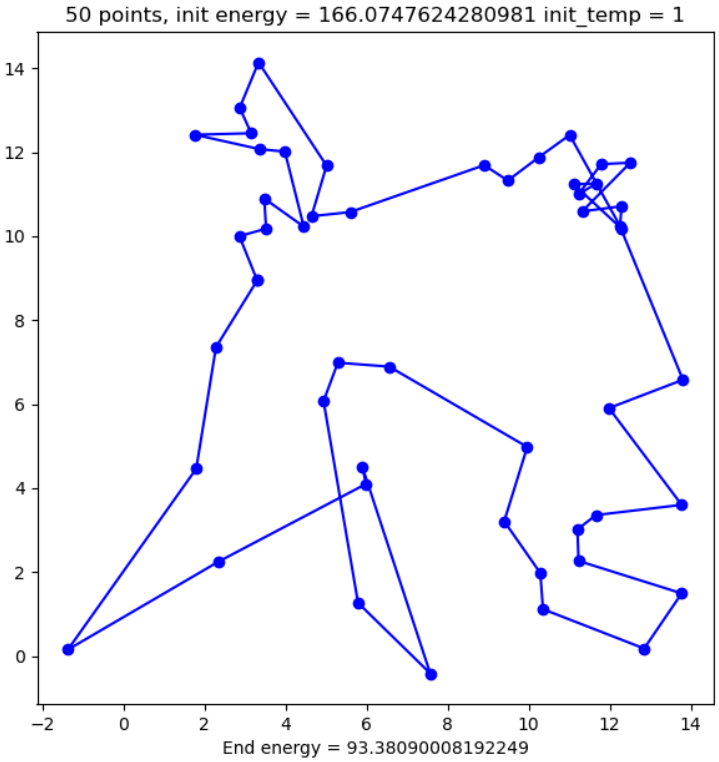





30 points, init_temp = 1
12
10
8
6
4
2
2
4
6
8
10
12
14
Start energy = 108.82395621151522

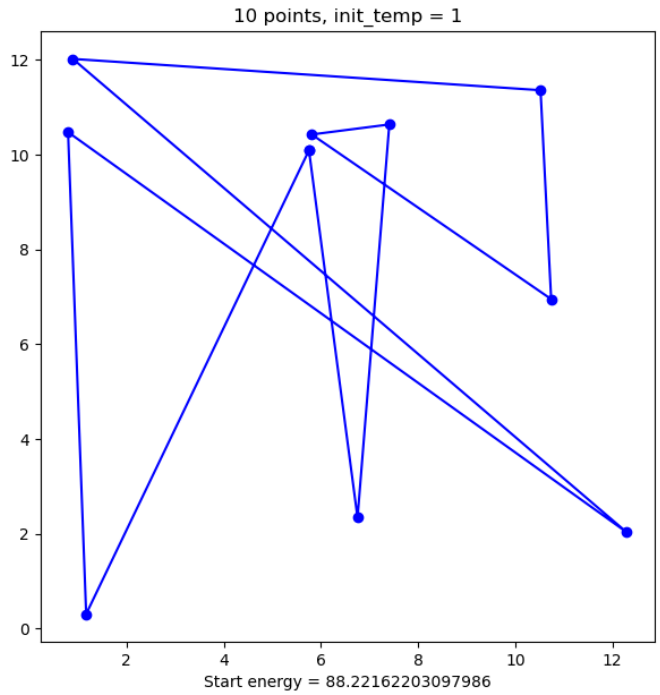

30 points, init energy = 108.82395621151522 init_temp = 1
12
10
8
6
4
2
2
4
6
8
10
12
14
End energy = 63.09527667044194

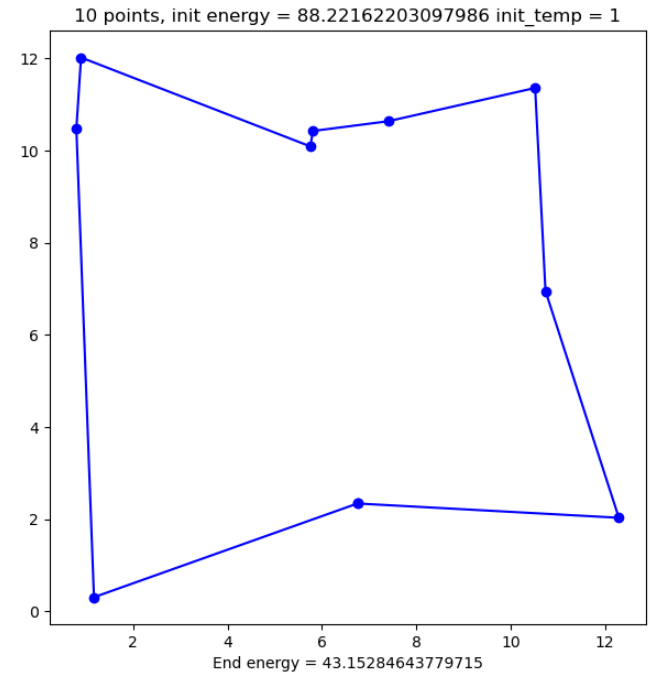

50 points, init_temp = 1
Start energy = 166.0747624280981

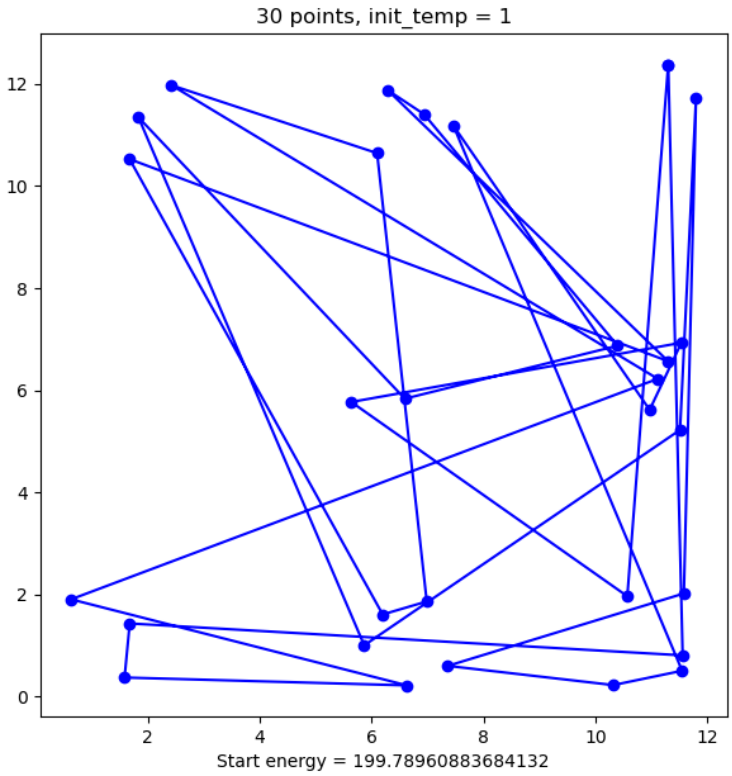

50 points, init energy = 166.0747624280981 init_temp = 1
End energy = 93.38090008192249

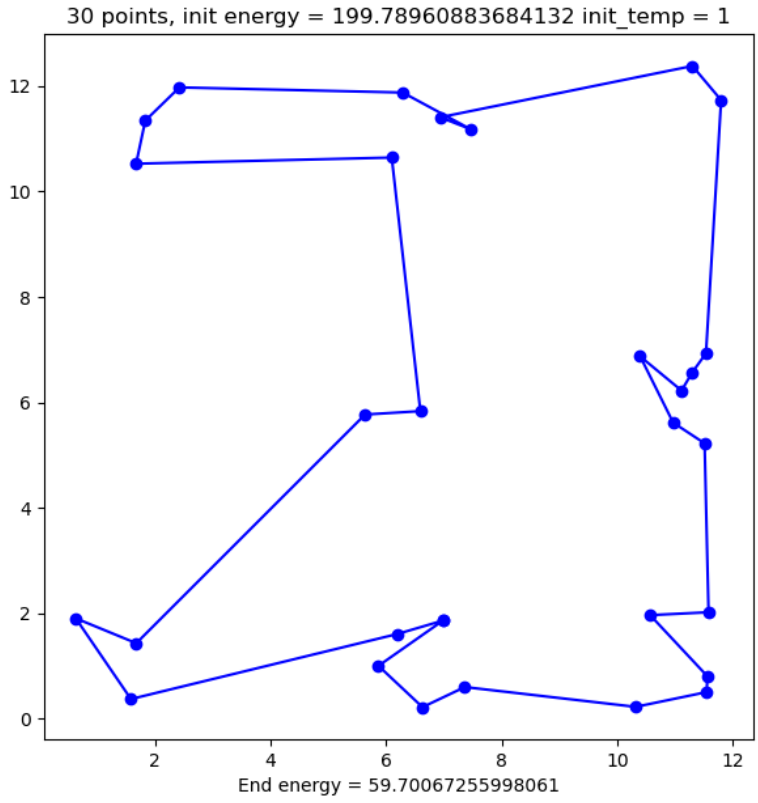
Dla drugiego sposobu generowania obserwujemy, że zmiany energii nie są już tak duże, jak w przypadku pierwszego sposobu. Drugi sposób tworzy układ, który jest bardziej zbliżony do optymalnego rozwiązania.

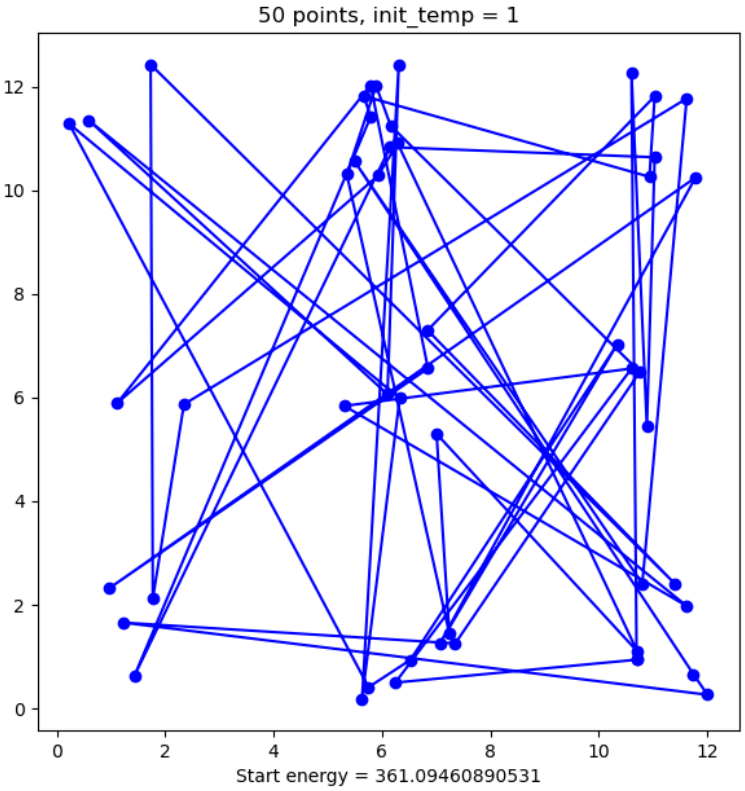
Generowanie z 9 odseparowanymi grupami punktów.

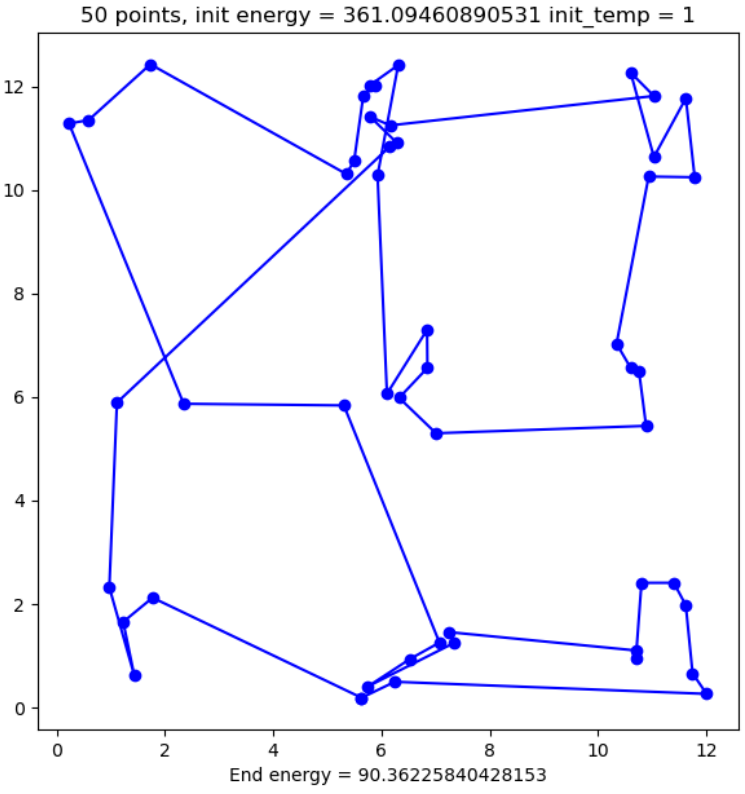





30 points, init_temp = 1
0
2
4
6
8
10
12
Start energy = 199.78960883684132


30 points, init energy = 199.78960883684132 init_temp = 1
0
2
4
6
8
10
12
End energy = 59.70067255998061


50 points, init_temp = 1
0
2
4
6
8
10
12
Start energy = 361.09460890531


50 points, init energy = 361.09460890531 init_temp = 1
0
2
4
6
8
10
12
End energy = 90.36225840428153

Dla ostatniego przypadku generowania punktów zmiany połączenia pomiędzy punktami mamy znaczne poprawienie wyniku. Jest to spowodowane tym, że losowo generowane punkty mają wiele połączeń pomiędzy grupami, które powodują dużą energię, po optymalizacji widzimy, że grupy są połączone tylko jedną lub dwoma krawędziami.

b) W tym podpunkcie badamy zależność zbieżności procesu od funkcji temperatury i sposobu generacji stanu sąsiedniego.

Różnica pomiędzy arbitrary i consecutive dla ustawień z początku sprawozdania:

```
Initial energy: 205.27520912391088, points: 25, initial temp: 1
End energy with consecutive func :  154.69752173294066
End energy with consecutive func :  128.74948989888352
End energy with consecutive func :  143.28683449194185
End energy with consecutive func :  119.35882549737485
End energy with consecutive func :  131.95892445085613
End energy with consecutive func :  129.70905324093226
End energy with consecutive func :  141.65810364754412
End energy with consecutive func :  112.69663308885727
End energy with consecutive func :  137.3866136217404
End energy with consecutive func :  121.53684329736157

End energy with arbitrary func :  63.732030587909385
End energy with arbitrary func :  69.89525411517829
End energy with arbitrary func :  62.61739213163894
End energy with arbitrary func :  66.04052669436014
End energy with arbitrary func :  65.981530193979
End energy with arbitrary func :  63.673669558348806
End energy with arbitrary func :  65.20017723470059
End energy with arbitrary func :  67.71468254627558
End energy with arbitrary func :  69.96010908747257
End energy with arbitrary func :  65.53793018886431
```

Widzimy, że dla arbitrary wyniki są niższe, niż dla consecutive.

Różnica pomiędzy różnymi funkcjami temperatury dla tych samych ustawień, co wcześniej, z generowaniem stanów sąsiednich arbitrary:

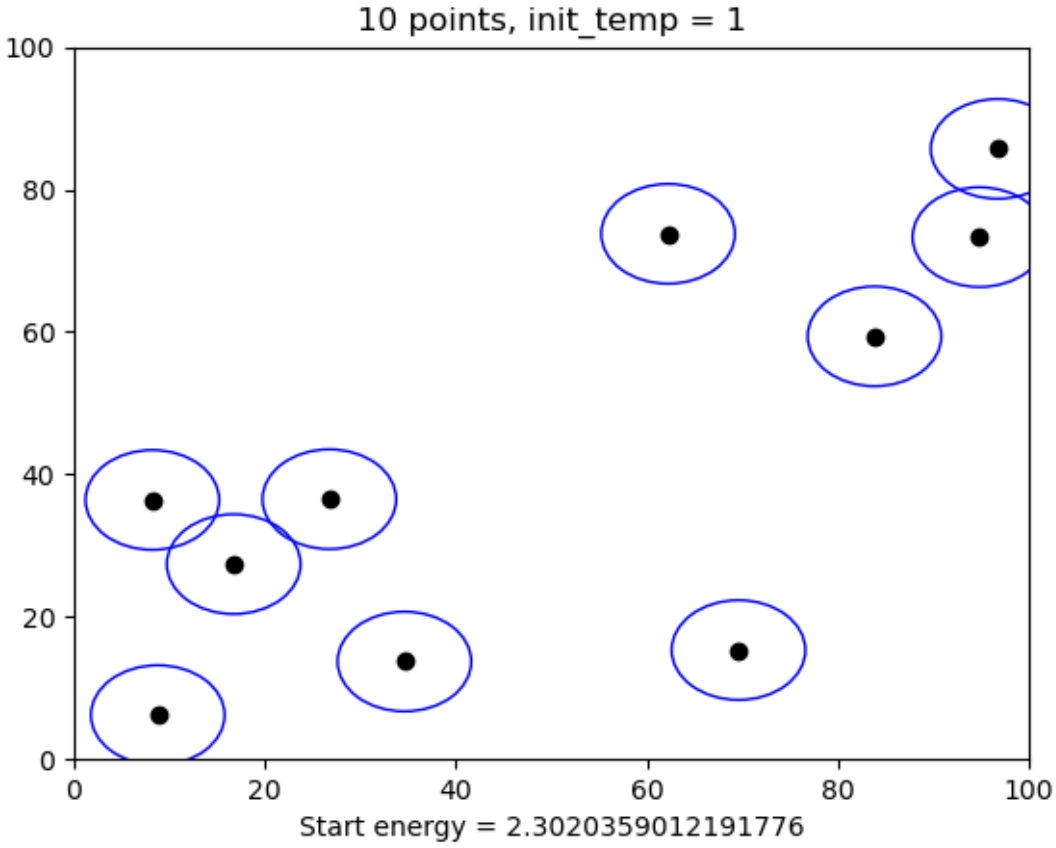
```
Initial energy: 214.58625695185236, points: 25, initial temp: 1
End energy with 1 - iteration / max_iteration func :  139.23196434033653
End energy with 1 - iteration / max_iteration func :  122.36548203292786
End energy with 1 - iteration / max_iteration func :  127.55610702085903
End energy with 1 - iteration / max_iteration func :  132.17024482765092
End energy with 1 - iteration / max_iteration func :  134.6504313204401
End energy with 1 - iteration / max_iteration func :  122.35661136143432
End energy with 1 - iteration / max_iteration func :  141.9074033581643
End energy with 1 - iteration / max_iteration func :  133.28389353147276
End energy with 1 - iteration / max_iteration func :  126.21824665601407
End energy with 1 - iteration / max_iteration func :  124.13925617889737

End energy with t_initial / iteration func :  132.74023976984958
End energy with t_initial / iteration func :  123.7220337247839
End energy with t_initial / iteration func :  121.48127528399733
End energy with t_initial / iteration func :  120.30963240705697
End energy with t_initial / iteration func :  124.22324372792212
End energy with t_initial / iteration func :  119.87346149950753
End energy with t_initial / iteration func :  121.96813748995847
End energy with t_initial / iteration func :  111.75953419338323
End energy with t_initial / iteration func :  134.87374805939427
End energy with t_initial / iteration func :  114.65517212188749

End energy with Boltzman func :  122.62560809792419
End energy with Boltzman func :  133.56256358442786
End energy with Boltzman func :  119.7319172952763
End energy with Boltzman func :  121.21309030902236
End energy with Boltzman func :  124.49556905803679
End energy with Boltzman func :  135.12389554955087
End energy with Boltzman func :  134.24519838491446
End energy with Boltzman func :  139.11130871977832
End energy with Boltzman func :  131.9202164041293
End energy with Boltzman func :  114.77527659861566
```

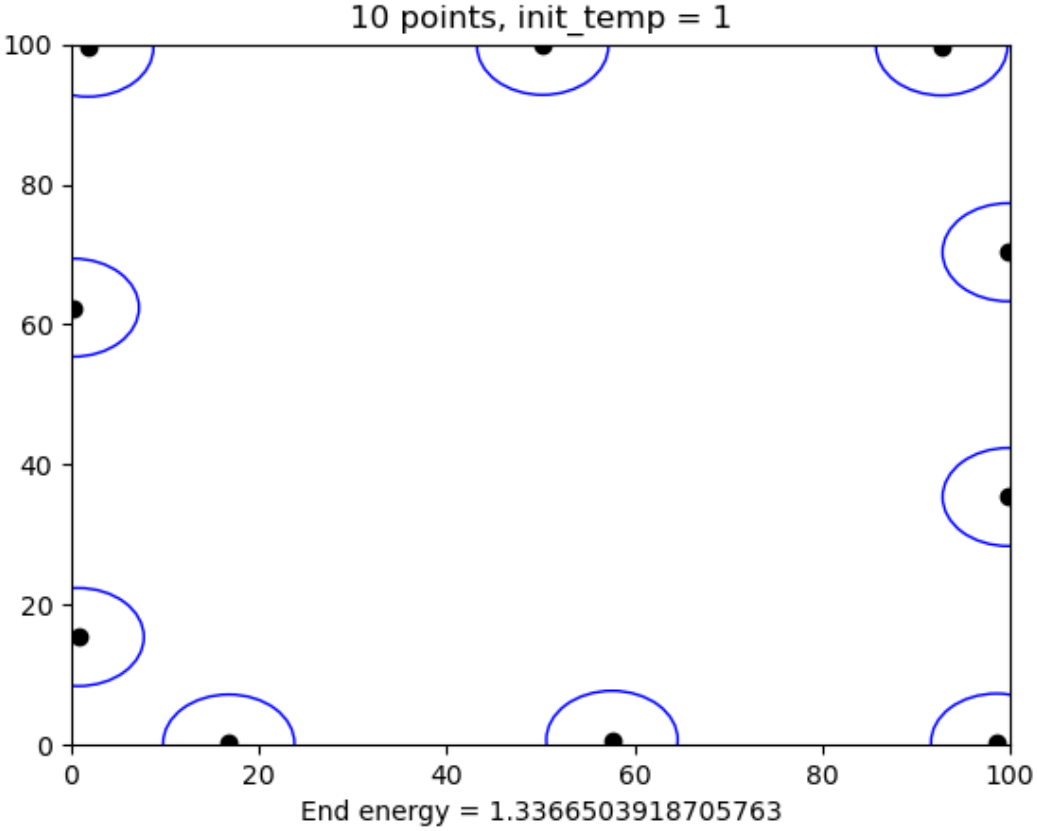
Pierwsza z funkcji nie dała żadnego wyniku poniżej 120, w przypadku drugiej i trzeciej wyniki są podobne.

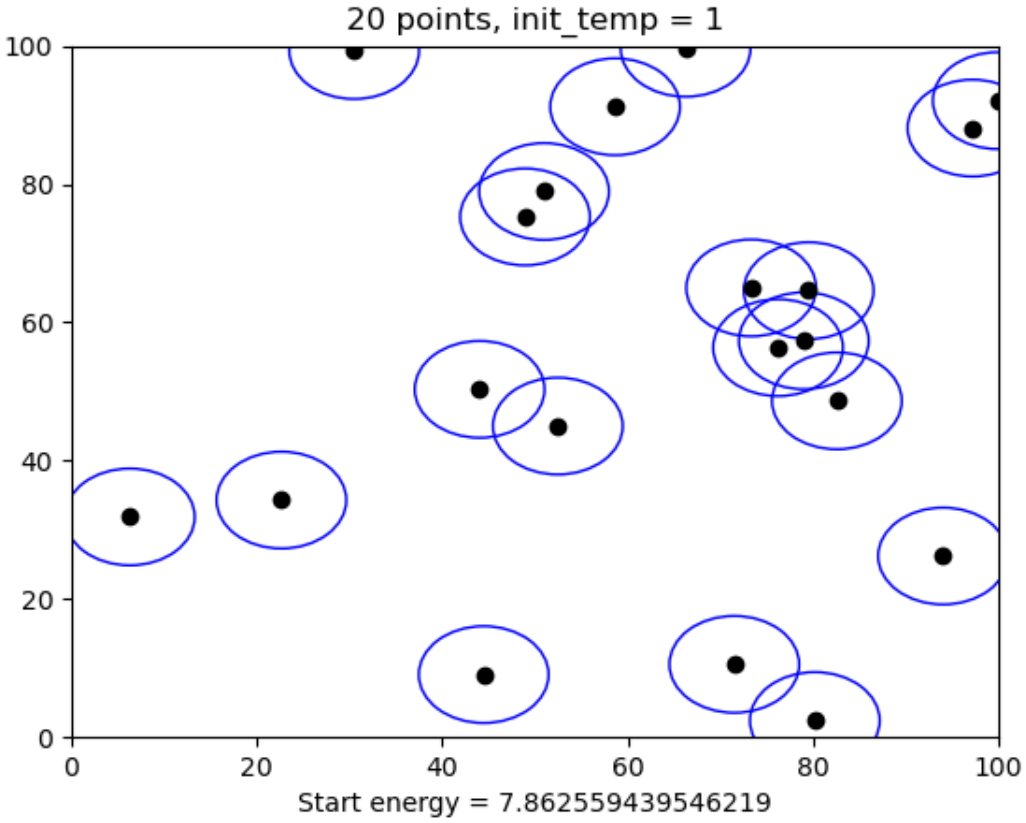
c) Wizualizacja procesu minimalizacji dla 18 punktów wygenerowanych 3 sposobem, czyli 9 odseparowanych punktów. Reszta ustawień jak na początku sprawozdania. Gif znajduje się w folderze pod nazwą „func_minim_visual.gif” lub pod linkiem https://gfycat.com/pl/foolishdampgemsbuck

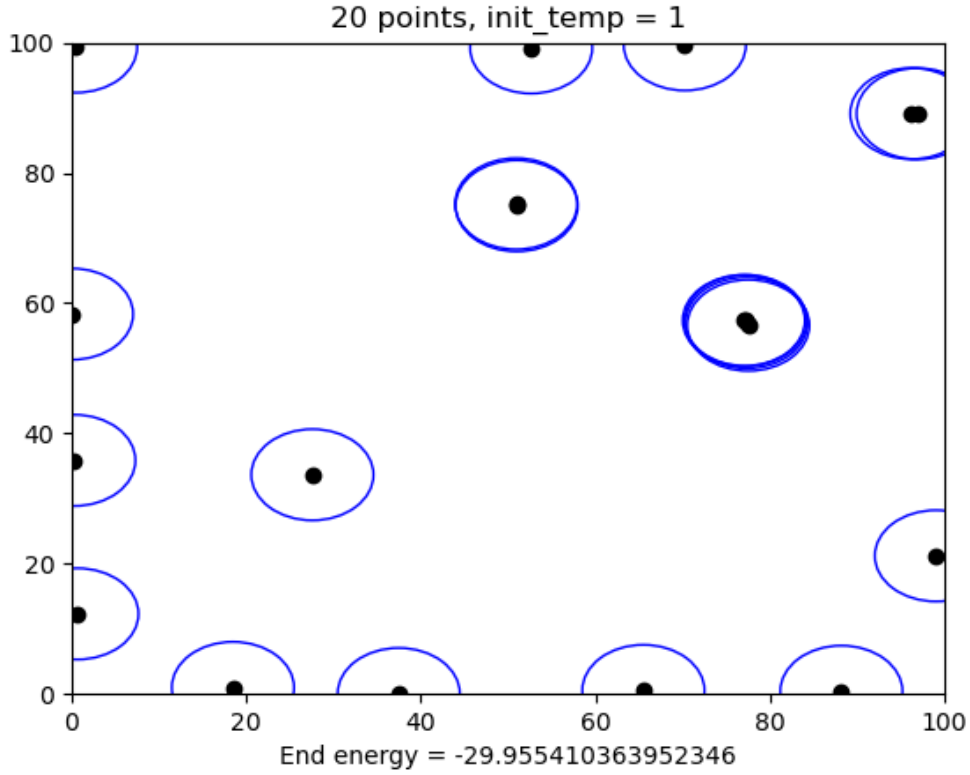
Zadanie 2 Obraz binarny

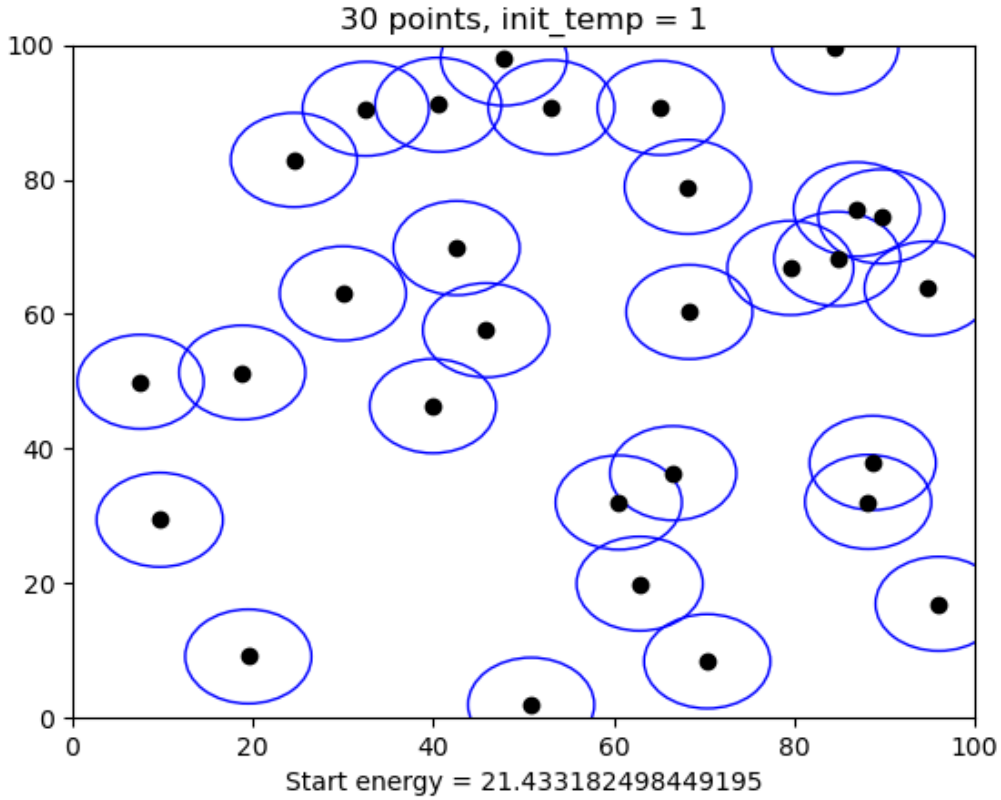
Wygenerowany został obraz 100x100 z gęstościami kolejno 0.1, 0.2, 0.3 i 0.4. Funkcją energii jest suma wartości PARAM * 1 / distance dla każdego punktu, gdzie PARAM to ustalany parametr, a distance to odległość pomiędzy dwoma punktami. Zakładamy, że punkt oddziaływuje z wszystkimi innymi. Jeśli punkt jest w odległości mniejszej niż pewna ustalona wartość FORCE_DIST (dla poniższych przykładów wynosi 7, wokół każdego punktu narysowany został okrąg symbolizujący tą odległość), to punkty się przyciągają, w przeciwnym wypadku odpychają. Stan sąsiedni generujemy w ten sposób, że wybieramy losowy punkt, dla niego losowy wektor przemieszczenia (o jeden w każdej z osi, np. o 1 w osi X i o -1 w osi Y). Jeśli punkt nie wychodzi poza obszar ani nie wchodzi na inny punkt, jest przesuwany.

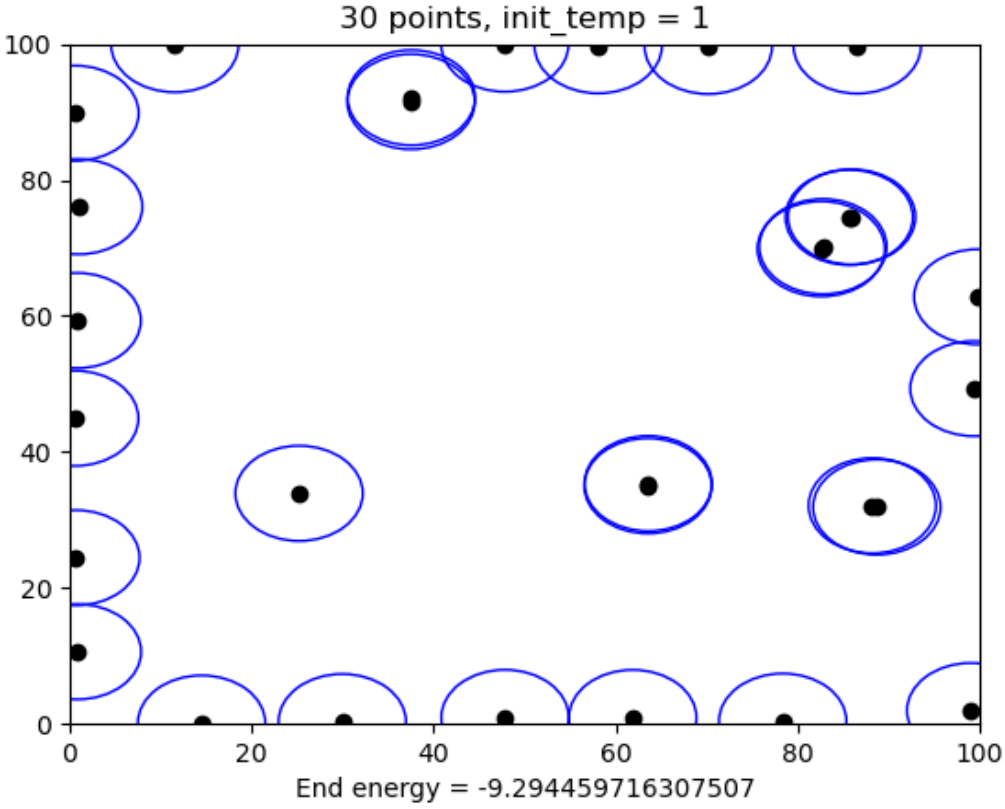

10 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
Start energy = 2.3020359012191776

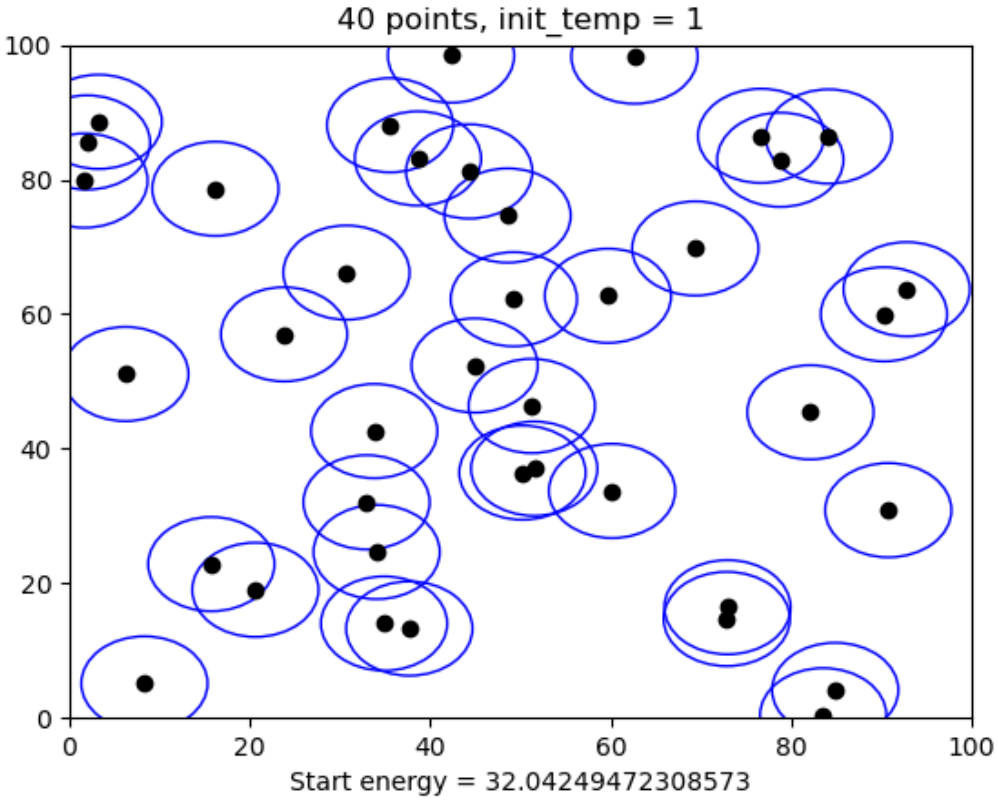

10 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
End energy = 1.3366503918705763

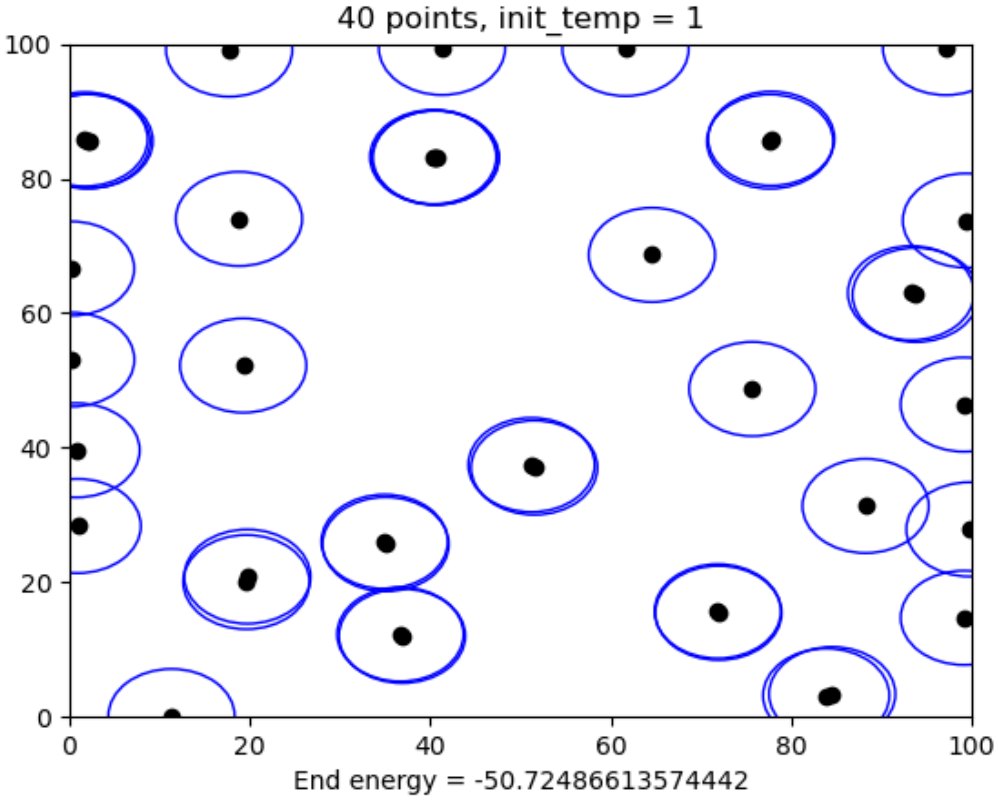

20 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
Start energy = 7.862559439546219


20 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
End energy = -29.955410363952346


30 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
Start energy = 21.433182498449195


30 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
End energy = -9.294459716307507

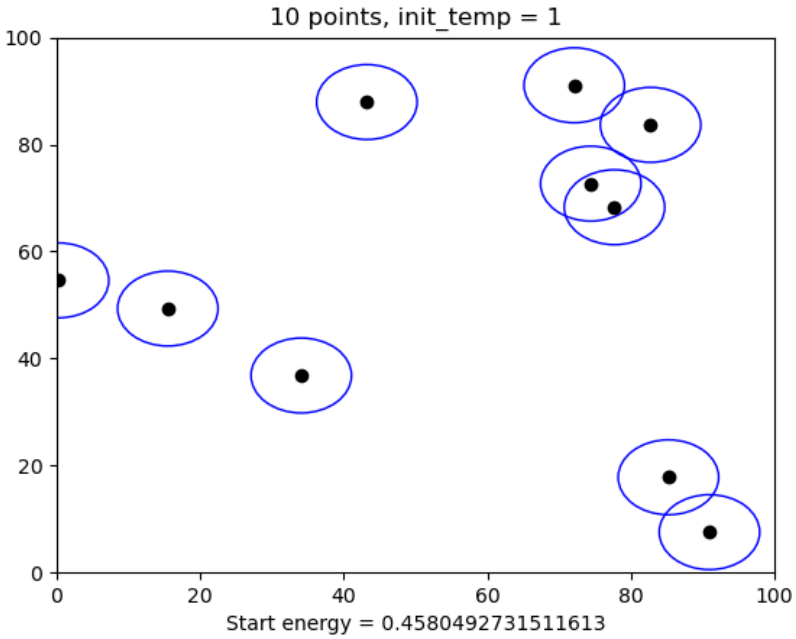

40 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
Start energy = 32.04249472308573

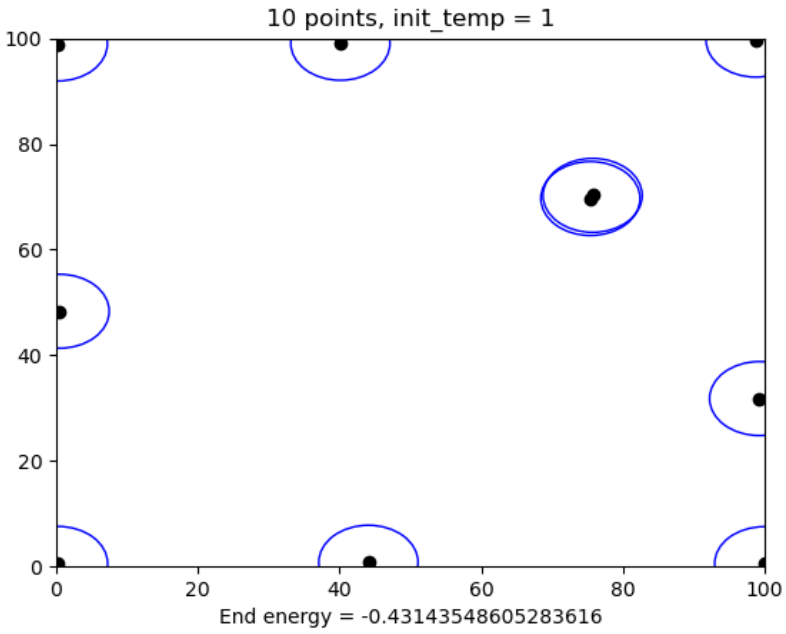

40 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
End energy = -50.72486613574442

Wyniki są bardzo interesujące. Punkty, które na początku były w odległości przyciągania się przybliżają się do siebie jeszcze bardziej, a punkty, które nie miały bliskich sąsiadów w większości pod koniec nadal pozostają same i zmierzają na granice obszaru.

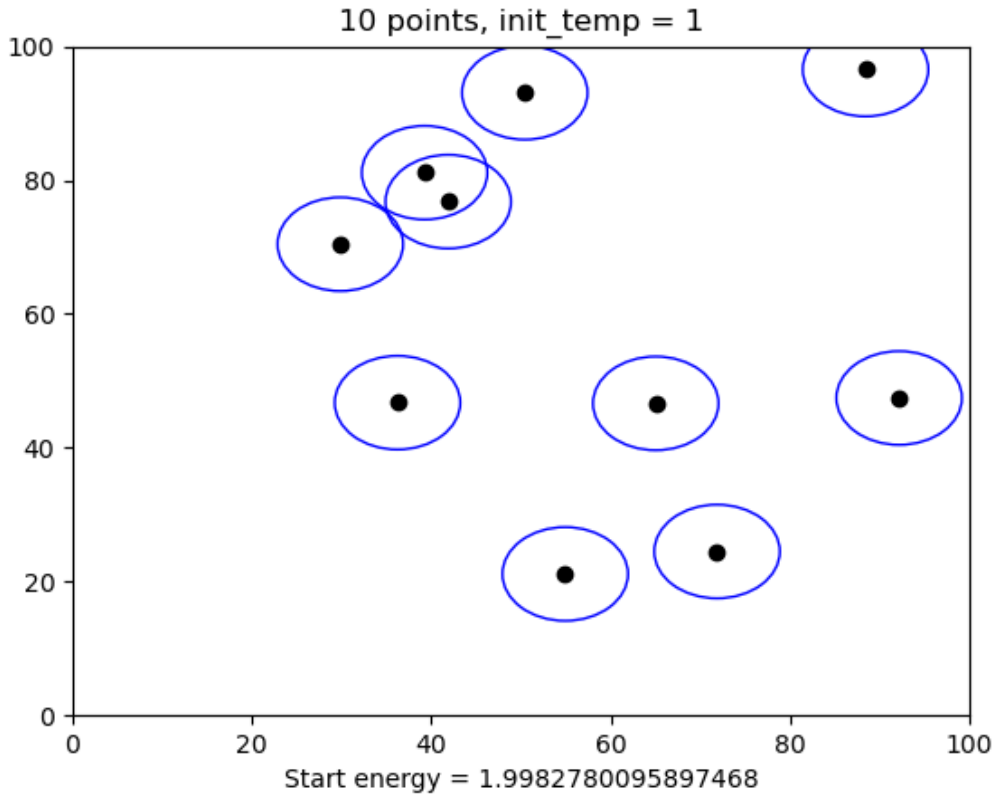
Wyniki dla 10 punktów i wartości parametru PARAM kolejno 0.25, 1, 4, 30.

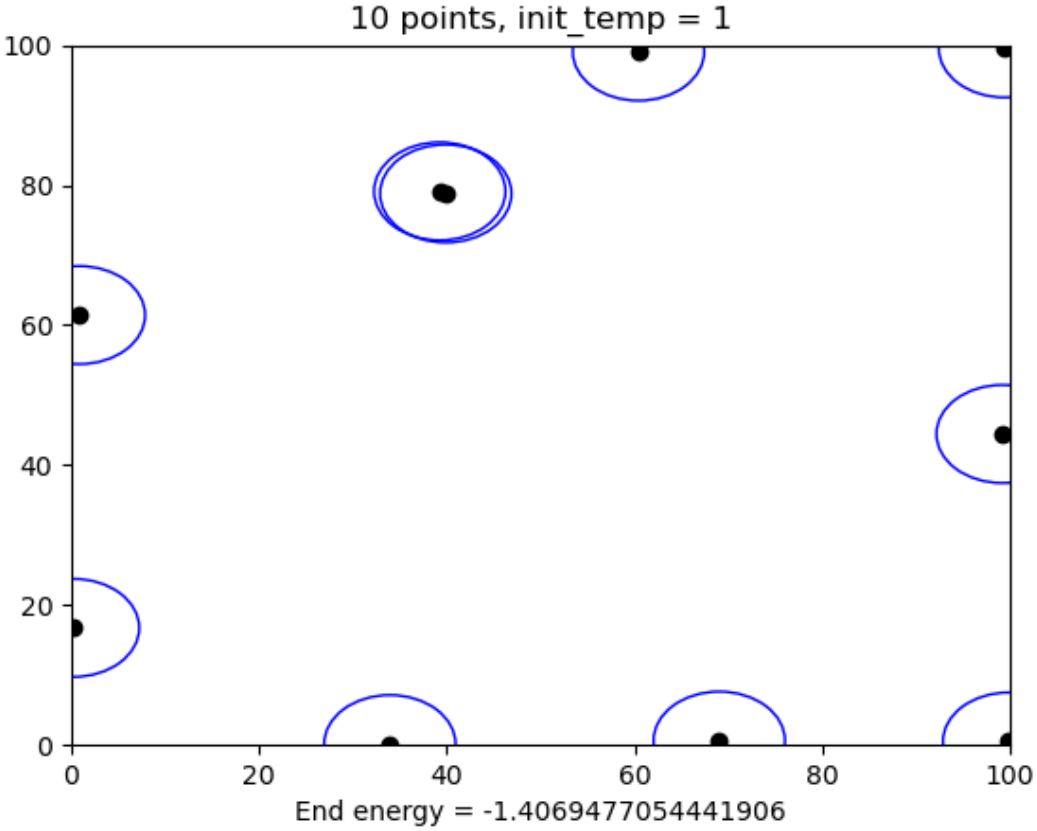
PARAM = 0.25



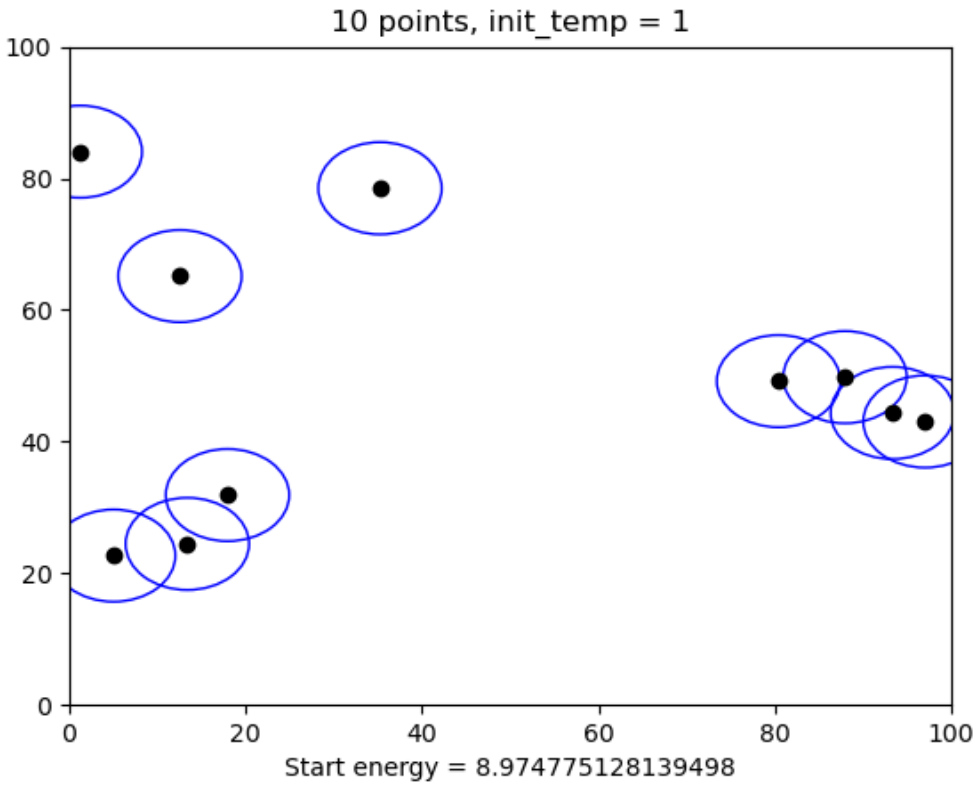


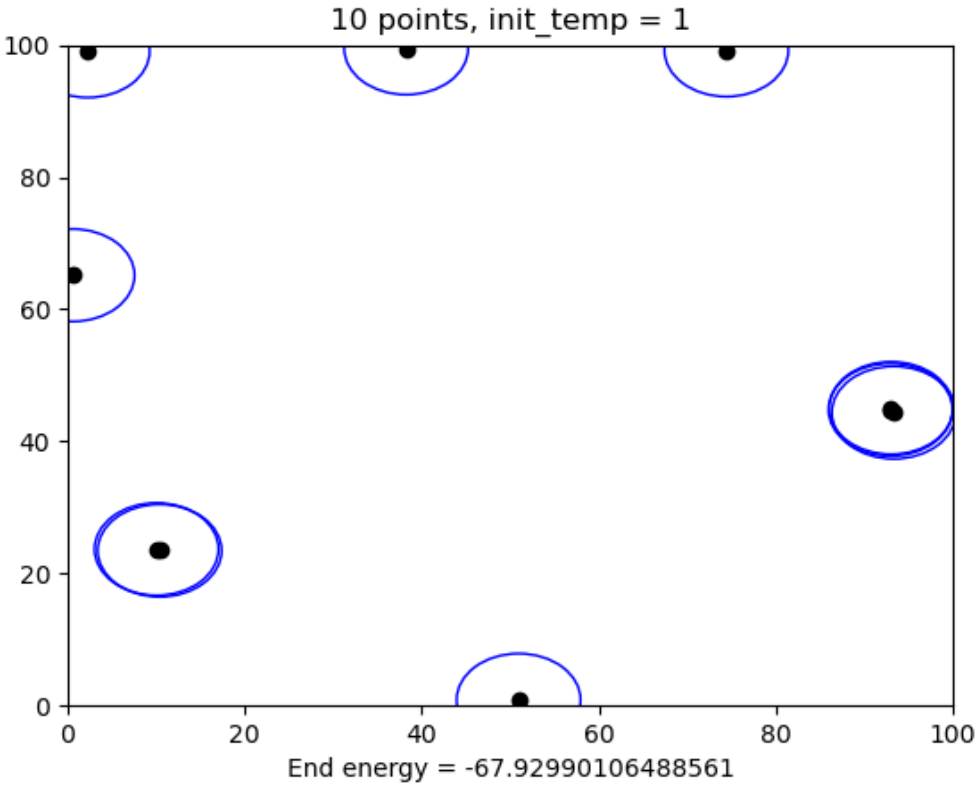
PARAM = 1


10 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
Start energy = 1.9982780095897468

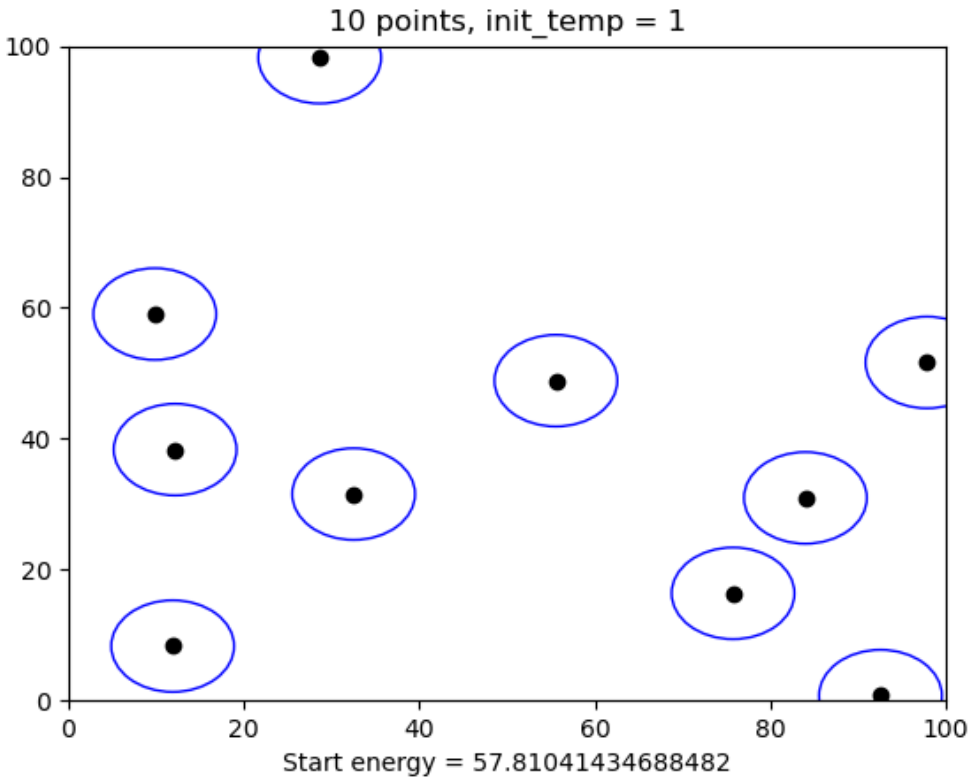

10 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
End energy = -1.4069477054441906

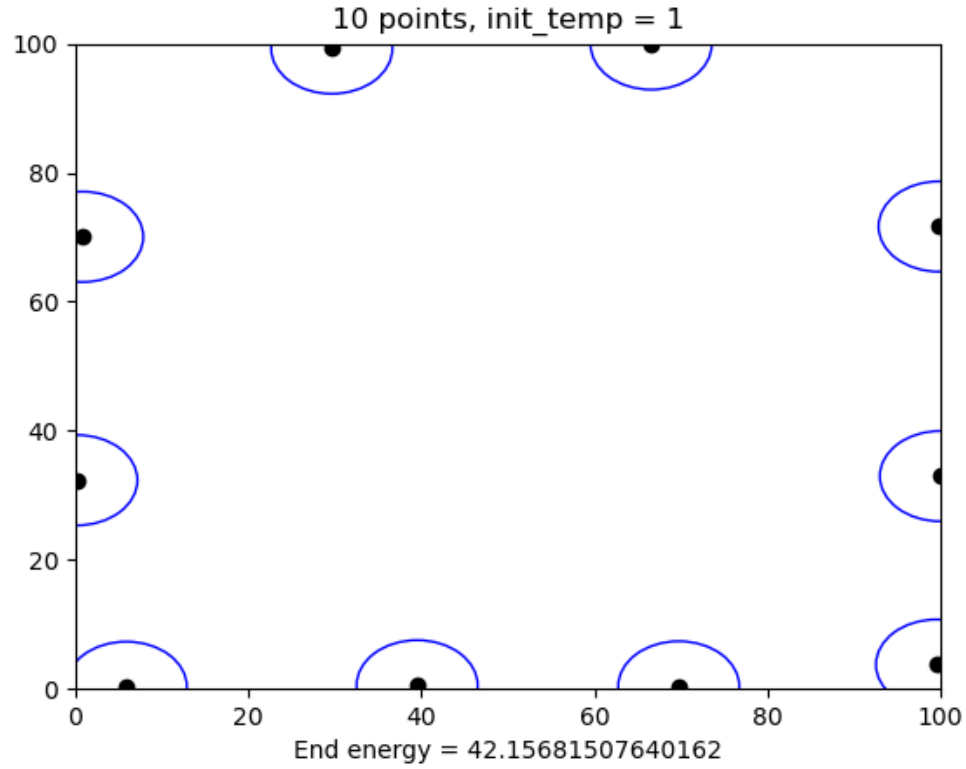
PARAM = 4


10 points, init_temp = 1
0
20
40
60
80
100
Start energy = 8.974775128139498


10 points, init_temp = 1
0
20
40
60
80
100
End energy = -67.92990106488561

PARAM = 30


10 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
Start energy = 57.81041434688482


10 points, init_temp = 1
100
80
60
40
20
0
0
20
40
60
80
100
End energy = 42.15681507640162

Zmiany funkcji energii nie wpływają zbytnio na zachowanie się punktów. Głównym czynnikiem jest początkowe rozłożenie punktów.

Różne funkcje temperatury:

```
Initial energy: 2.375826508816965, points: 10, initial temp: 1
End energy with 1 - iteration / max_iteration func :  -31.39696249073995
End energy with 1 - iteration / max_iteration func :  -21.381817376891412
End energy with 1 - iteration / max_iteration func :  -25.046639341254906
End energy with 1 - iteration / max_iteration func :  -15.782451963373799
End energy with 1 - iteration / max_iteration func :  -9.537358766313211
End energy with 1 - iteration / max_iteration func :  -38.34352302682058
End energy with 1 - iteration / max_iteration func :  -17.154862402508865
End energy with 1 - iteration / max_iteration func :  -14.831088136298119
End energy with 1 - iteration / max_iteration func :  -27.272283762241
End energy with 1 - iteration / max_iteration func :  -23.033406755394147

End energy with t_initial / iteration func :  -1.3562885522010766
End energy with t_initial / iteration func :  -7.5930023114113165
End energy with t_initial / iteration func :  -7.504715668600327
End energy with t_initial / iteration func :  -7.501782858758616
End energy with t_initial / iteration func :  -7.508885591757131
End energy with t_initial / iteration func :  -7.540233169706802
End energy with t_initial / iteration func :  -1.3244213372574485
End energy with t_initial / iteration func :  -1.3822076325799475
End energy with t_initial / iteration func :  -7.515087151716843
End energy with t_initial / iteration func :  -1.3566728416423675

End energy with Boltzman func :  -11.223226910376095
End energy with Boltzman func :  -12.002422289316781
End energy with Boltzman func :  -23.459497981623187
End energy with Boltzman func :  -21.613607380500675
End energy with Boltzman func :  -11.101140576630964
End energy with Boltzman func :  -10.414301837954506
End energy with Boltzman func :  -12.303155431510945
End energy with Boltzman func :  -14.638133632926813
End energy with Boltzman func :  -25.467807963343546
End energy with Boltzman func :  -11.881705761926984
```

Druga funkcja temperatury nie powoduje takiego zbliżania się punktów do siebie, jak funkcje pierwsza i druga, pomiędzy którymi nie ma większych różnic.

Zadanie 3 Sudoku

Funkcja energii jest realizowana w ten sposób, że minimalną energią (rozwiązanie sudoku) jest -27, za każdą kolumnę, wiersz i blok 3x3, w którym nie powtarza się żadna cyfra otrzymujemy -1 energii. Plansza jest wczytywana z pliku ssv, niewypełnione pola zostały oznaczone zerami. Następnie wszystkie kwadraty są wypełniane unikalnymi liczbami tak, aby liczba jedynek, dwójek itd. w całej planszy wynosiła 9. Stan sąsiedni generujemy w ten sposób, że losujemy kwadrat i w nim zamieniamy dwie cyfry miejscami. Wypełnione cyfry mają przedrostek '_'.

Rozwiązanie dla „sudoku2.ssv".

```
0 4 0 0 1 0 3 0 0
0 1 0 6 0 0 5 0 0
0 0 7 0 0 5 0 0 6
2 0 0 0 0 0 0 0 8
7 3 0 0 9 8 0 6 5
8 6 0 3 2 4 7 0 0
0 8 0 4 6 7 9 5 0
0 0 9 0 0 0 0 7 0
4 0 2 8 0 9 6 1 0
```

Niewypełnione sudoku, wczytane z pliku.

```
---------------------------
|_2  4 _5|_2  1 _8| 3 _4 _7|
|_3  1 _9| 6 _4 _7| 5 _9 _2|
|_6 _8  7|_9 _3  5|_8 _1  6|
---------------------------
| 2 _4 _5|_6 _7 _1|_2 _1  8|
| 7  3 _1|_5  9  8|_3  6  5|
| 8  6 _9| 3  2  4| 7 _4 _9|
---------------------------
|_6  8 _1| 4  6  7| 9  5 _2|
|_3 _7  9|_2 _5 _1|_4  7 _3|
| 4 _5  2| 8 _3  9| 6  1 _8|
---------------------------
```

Sudoku wypełnione unikalnymi cyframi, przed procesem minimalizacji

```
---------------------------
|_5  4 _6|_7  1 _2| 3 _8 _9|
|_9  1 _8| 6 _4 _3| 5 _2 _7|
|_3 _2  7|_9 _8  5|_1 _4  6|
---------------------------
| 2 _9 _1|_5 _7 _6|_4 _3  8|
| 7  3 _4|_1  9  8|_2  6  5|
| 8  6 _5| 3  2  4| 7 _9 _1|
---------------------------
|_1  8 _3| 4  6  7| 9  5 _2|
|_6 _5  9|_2 _3 _1|_8  7 _4|
| 4 _7  2| 8 _5  9| 6  1 _3|
---------------------------
```

Rozwiązane sudoku po procesie minimalizacji

10000 iteracji

Dla drugiego przykładu, „sudoku1.ssv” nie otrzymujemy całkowitego rozwiązania.

```
0 4 8 0 0 0 0 0 1
5 0 0 8 0 0 4 0 7
0 0 0 3 0 0 5 0 0
0 2 3 0 1 0 0 0 0
0 6 0 0 4 0 0 9 0
0 0 0 0 5 0 7 1 0
0 0 9 0 0 5 0 0 0
8 0 7 0 0 2 0 0 4
2 0 0 0 0 0 9 5 0
```

Niewypełnione sudoku, wczytane z pliku.

```
---------------------------
|_3  4  8|_1 _4 _2|_9 _6  1|
| 5 _7 _1| 8 _6 _7| 4 _8  7|
|_6 _9 _2| 3 _9 _5| 5 _3 _2|
---------------------------
|_7  2  3|_3  1 _8|_6 _5 _8|
|_8  6 _5|_2  4 _9|_3  9 _4|
|_4 _9 _1|_6  5 _7| 7  1 _2|
---------------------------
|_4 _1  9|_4 _3  5|_6 _8 _3|
| 8 _5  7|_6 _8  2|_1 _2  4|
| 2 _3 _6|_7 _1 _9| 9  5 _7|
---------------------------
```

Sudoku wypełnione unikalnymi cyframi, przed

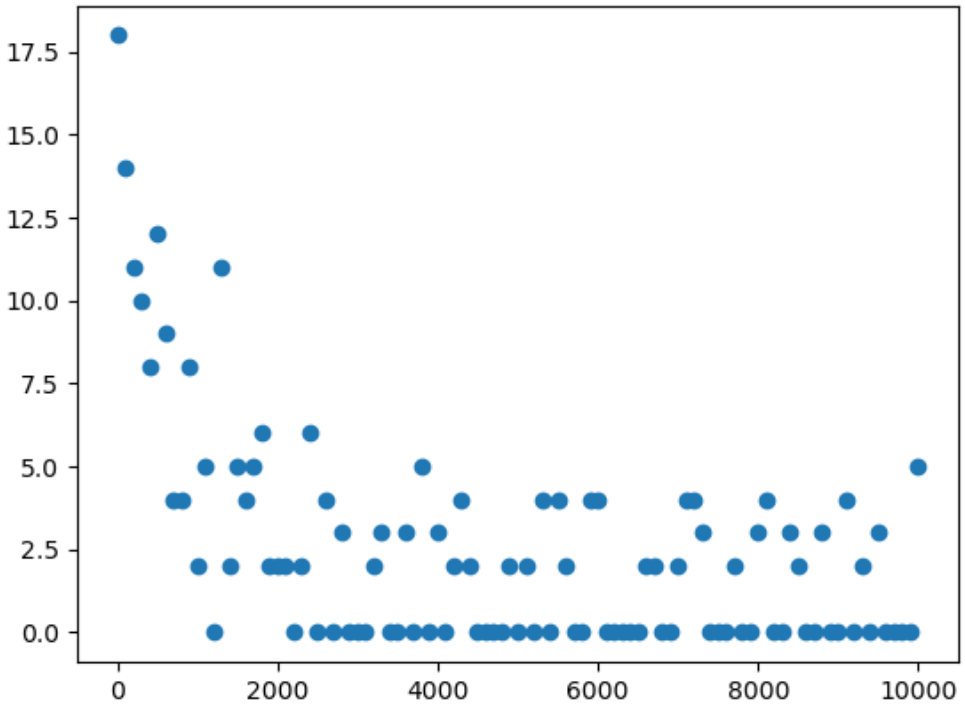
procesem minimalizacji

```
-------------------------
|_3  4  8|_5 _9 _7|_2 _6  1|
| 5 _9 _6| 8 _2 _1| 4 _3  7|
|_2 _7 _1| 3 _6 _4| 5 _8 _9|
-------------------------
|_5  2  3|_7  1 _9|_8 _4 _6|
|_1  6 _7|_2  4 _8|_3  9 _5|
|_4 _8 _9|_6  5 _3| 7  1 _2|
-------------------------
|_4 _3  9|_1 _8  5|_1 _7 _3|
| 8 _5  7|_9 _3  2|_6 _2  4|
| 2 _1 _6|_4 _7 _6| 9  5 _8|
-------------------------
```

Sudoku po procesie minimalizacji, lecz końcowa energia wynosi -22

10000 iteracji

Zatem odpowiadając na pytanie z zadania, program nie jest w stanie rozwiązać każdego sudoku.



Wykres zależności różnicy energii poprawnego rozwiązania i rozwiązania sudoku po procesie minimalizacji od liczby iteracji dla „sudoku2.ssv”

Z wykresu możemy odczytać, że po ok 3000 iteracjach nie zyskujemy na poprawie wyniku. Nadal występują niekompletne rozwiązania, nawet dla 10000 tysięcy iteracji, lecz powyżej 3000 iteracji różnica energii nie jest większa niż 6. Również gęstość niekompletnych rozwiązań się nie zmniejsza, tzn. w przedziale od 3000 do 6500 i od 6500 do 10000 iteracji jest mniej więcej tyle samo niekompletnych rozwiązań.