

Metody obliczeniowe w nauce i technice 2

Tomasz Zawadzki

17 kwietnia 2019

Zadanie 1. Problem wędrującego komiwojażera

Algorytm znajdujący optymalne rozwiązanie problemu wędrującego komiwojażera przy użyciu symulowanego wyżarzania został zaimplementowany w języku Python. Funkcją energii układu jest długość ścieżki, natomiast funkcją odległości dwóch punktów jest metryka euklidesowa. Rozważane są dwie metody generowania kolejnego stanu:

- *consecutive swap* – polegająca na zamianie kolejności dwóch kolejnych węzłów na ścieżce,
- *arbitrary swap* – polegająca na zamianie kolejności dwóch dowolnych węzłów na ścieżce.

Metoda *consecutive swap* znajduje optymalne rozwiązanie w znacznie mniejszej liczbie iteracji niż metoda *arbitrary swap*.

Schemat chłodzenia jest określony geometryczną funkcją temperatury postaci

$$T(n) = T_0 a^n, \quad a \in (0, 1), \quad (1)$$

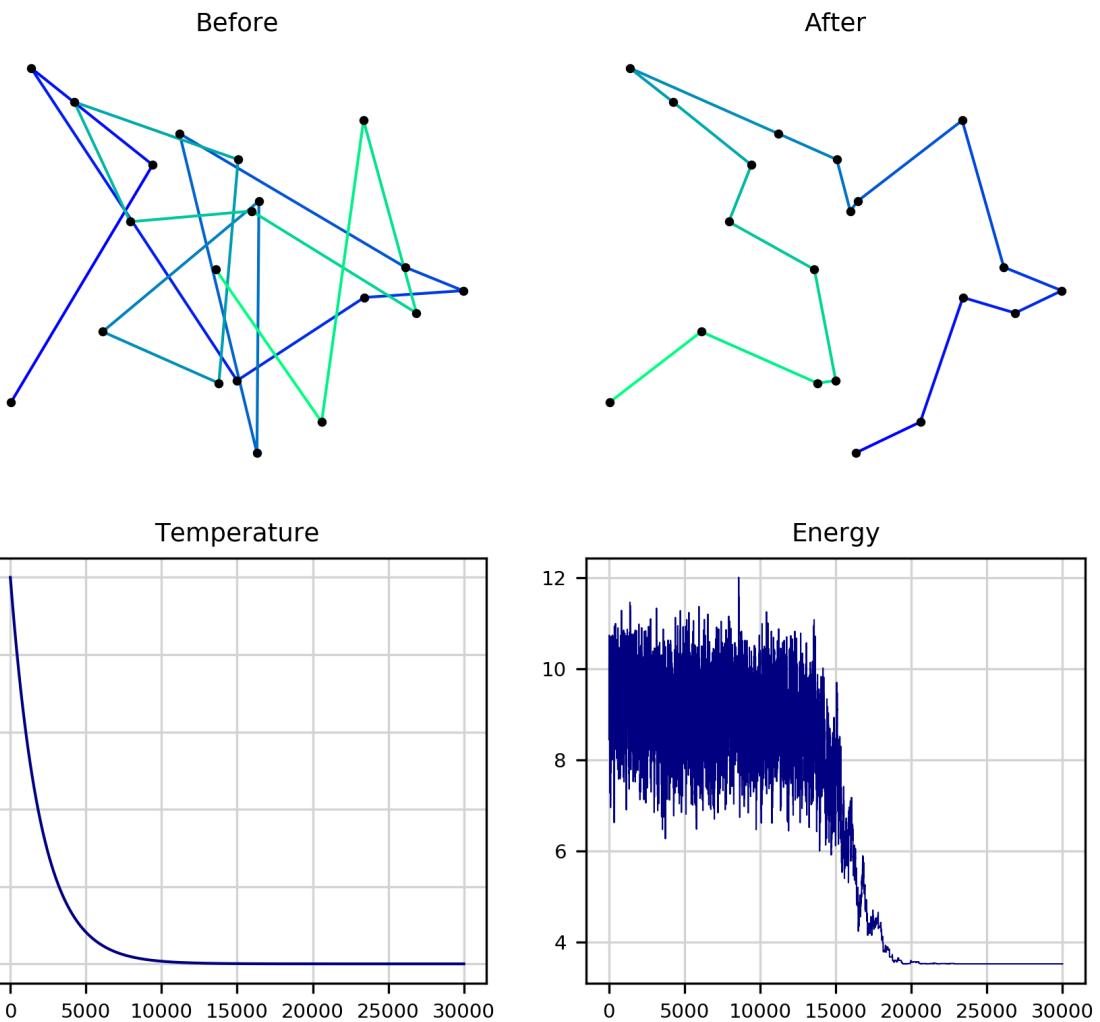
Współczynniki T_0 oraz a zostały znalezione empirycznie na podstawie analizy rezultatów działania programu. Prawdopodobieństwo przyjęcia stanu o wyższej energii jest określone wzorem

$$p = \exp\left(\frac{-\Delta}{T}\right), \quad \Delta = E' - E \quad (2)$$

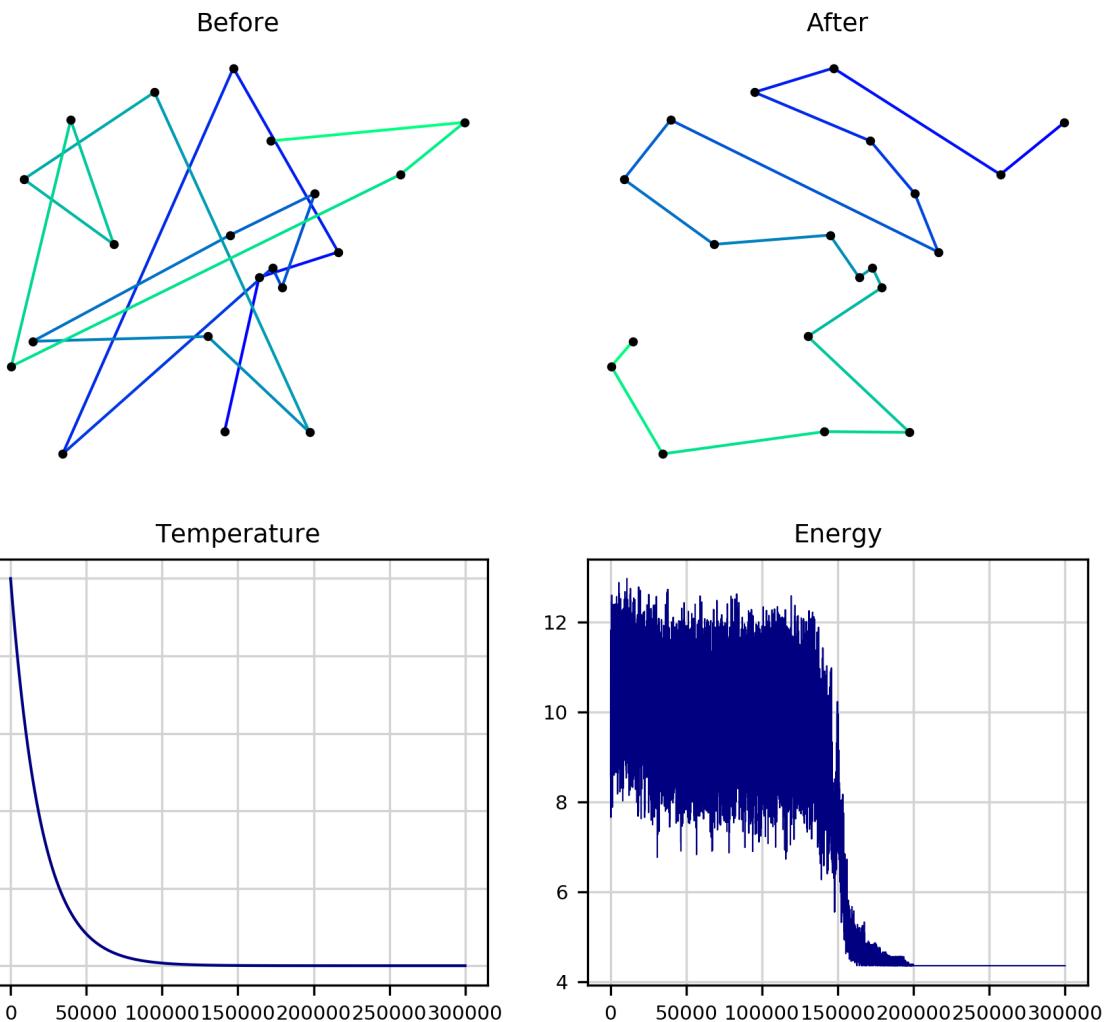
Dla rozkładów składających się z grup punktów rozważana była również alternatywna funkcja odległości między punktami uwzględniająca również dodatkowo odległość pomiędzy grupami, do których przynależą oba punkty.

$$d((x_1, y_1), (x_2, y_2)) = \mu \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + (\lfloor x_2 \rfloor - \lfloor x_1 \rfloor) + (\lfloor y_2 \rfloor - \lfloor y_1 \rfloor). \quad (3)$$

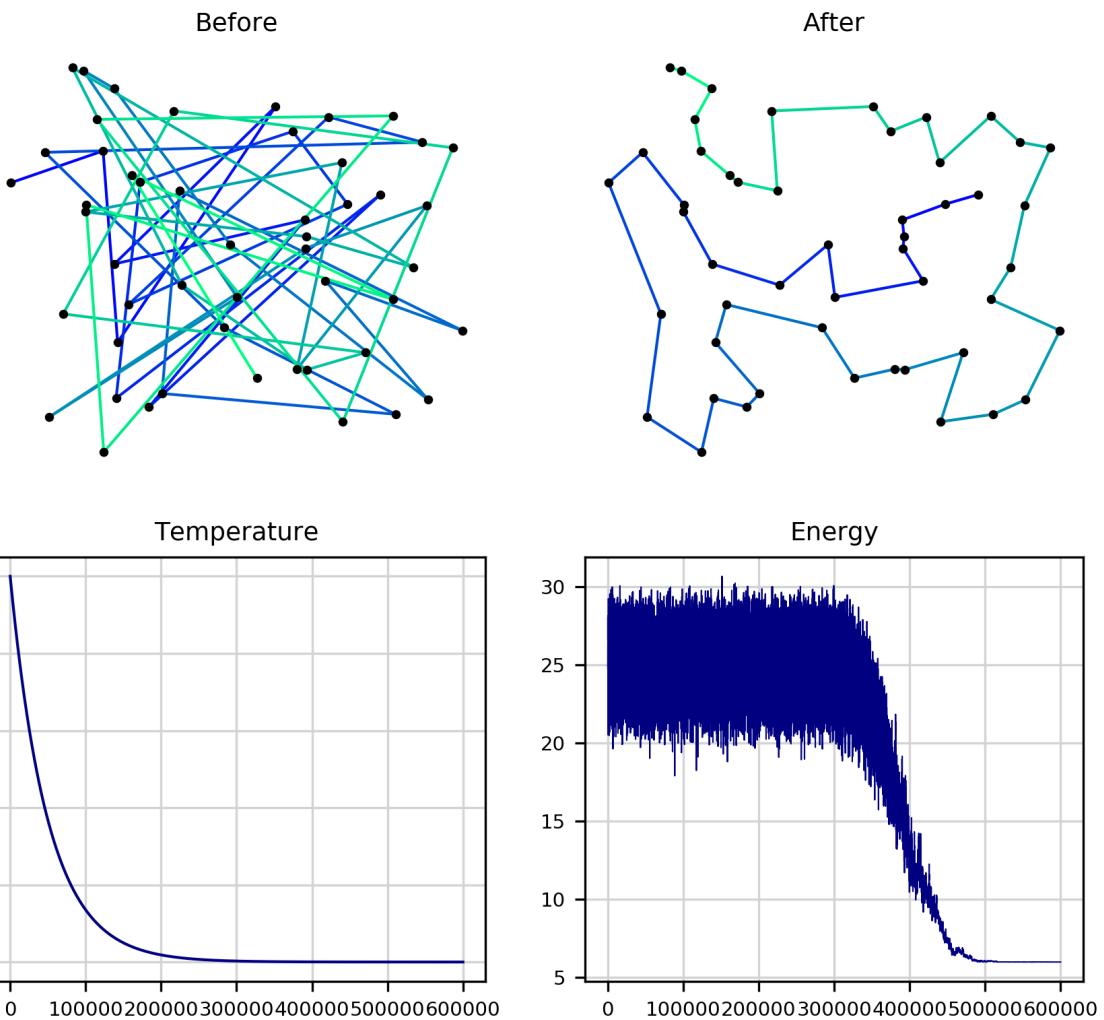
W przypadku rozkładów z różnymi grupami parametrów lepszym sposobem optymalizowania długości trasy mogłoby być hybrydowe rozwiązywanie problemu polegające na znalezieniu trasy między wszystkimi grupami, a następnie między wszystkimi punktami w każdej grupie osobno.



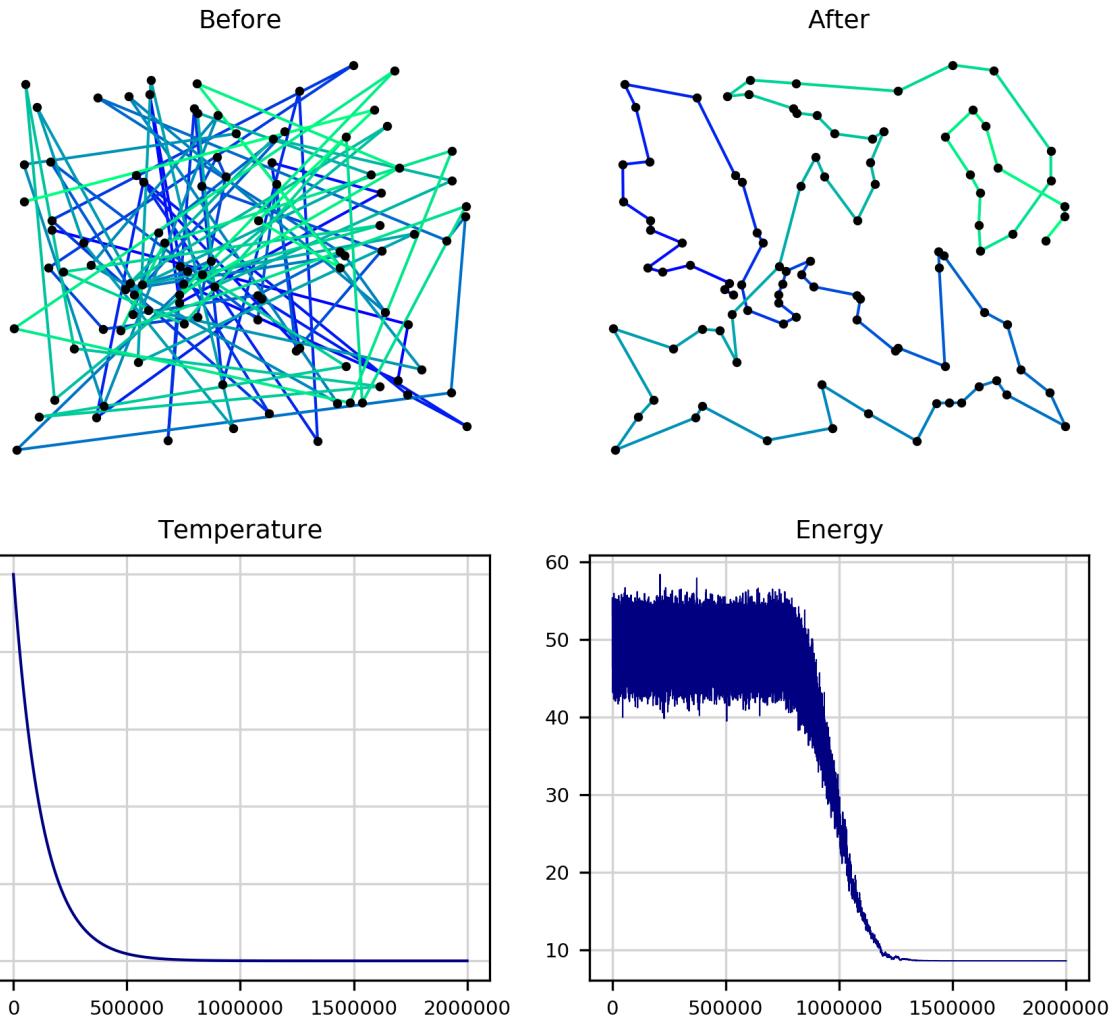
Rys. 1. $n = 20$, rozkład jednostajny, *arbitrary swap*, $T(n) = (0.9995)^n$, $I_{\max} = 3 \cdot 10^4$



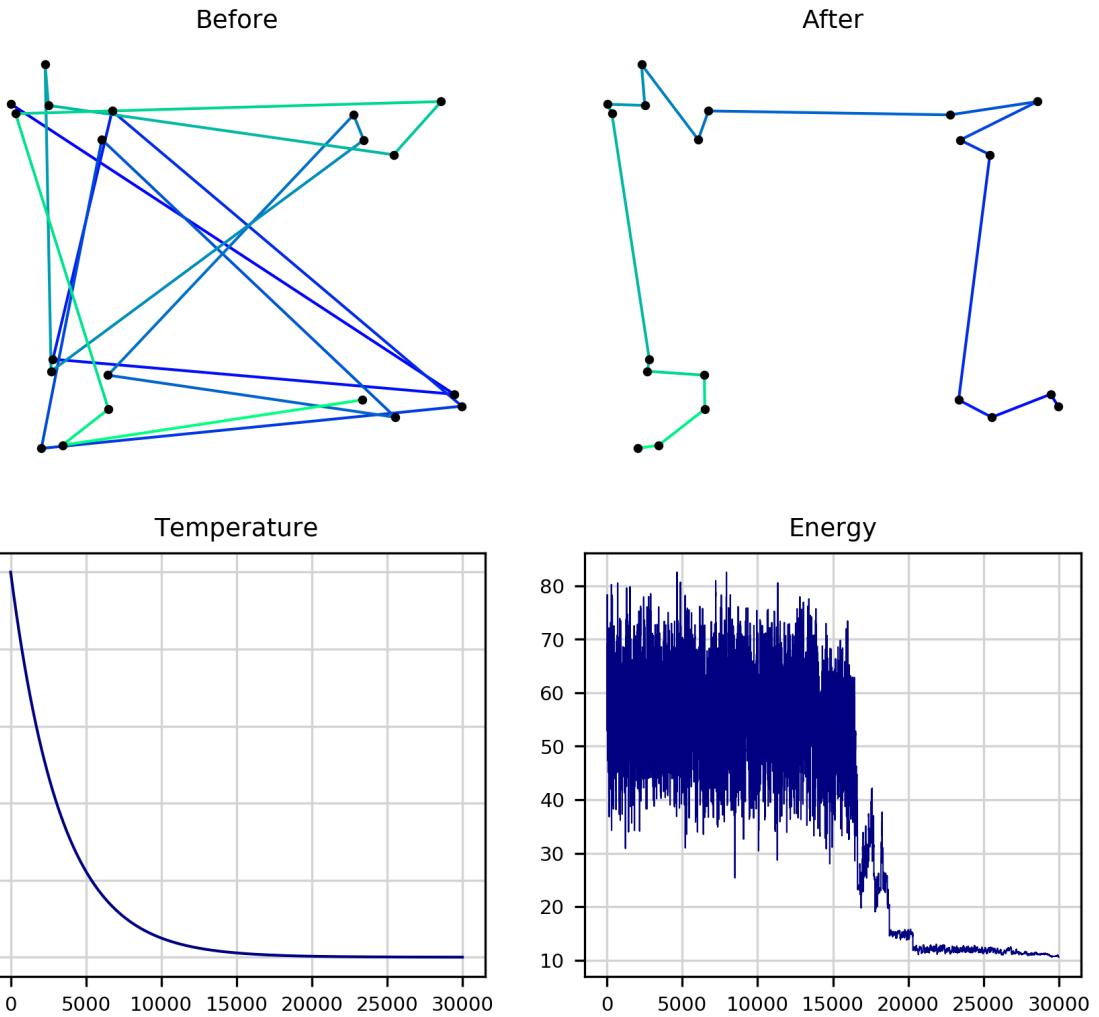
Rys. 2. $n = 20$, rozkład jednostajny, consecutive swap, $T(n) = (0.99995)^n$, $I_{\max} = 3 \cdot 10^5$



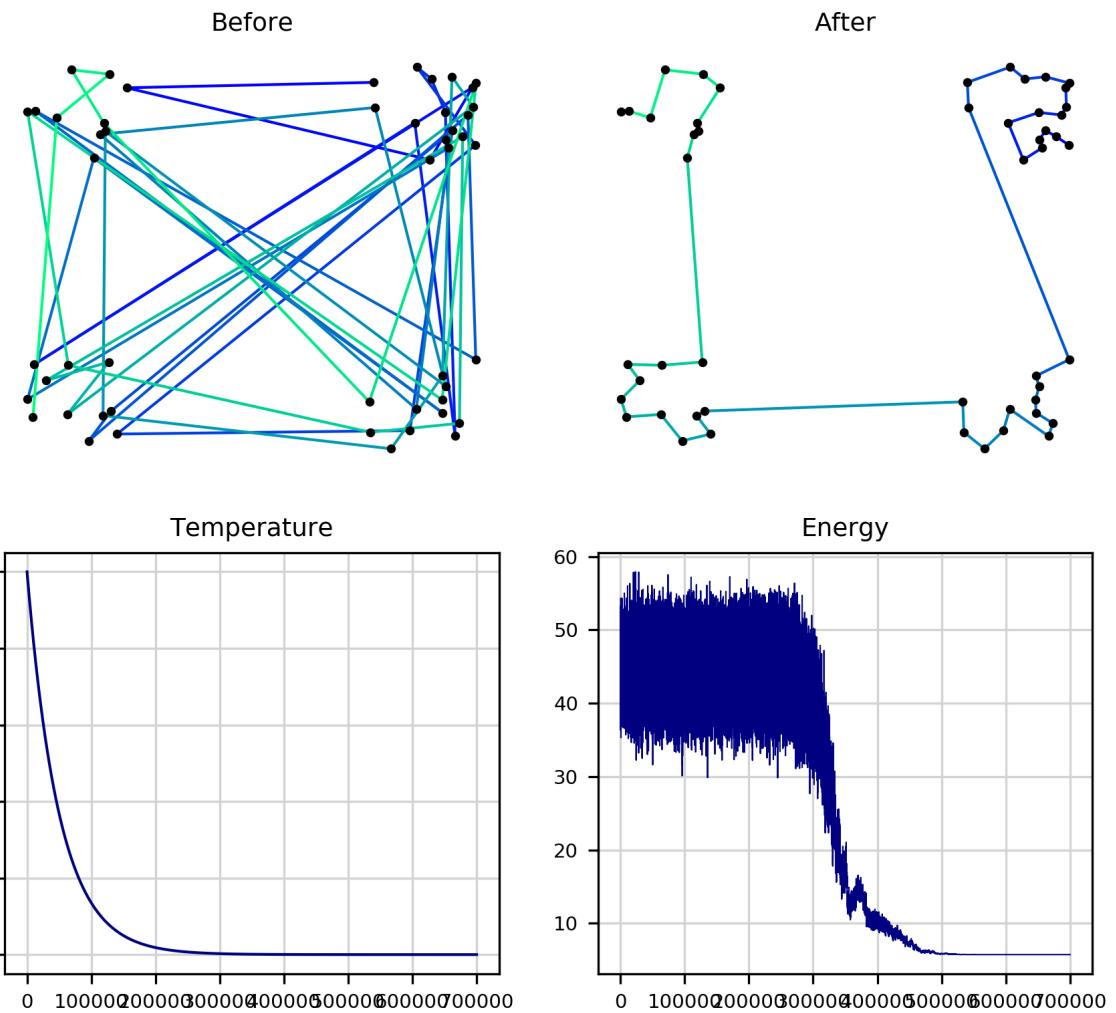
Rys. 3. $n = 50$, rozkład jednostajny, *arbitrary swap*, $T(n) = (0.99998)^n$, $I_{\max} = 6 \cdot 10^5$



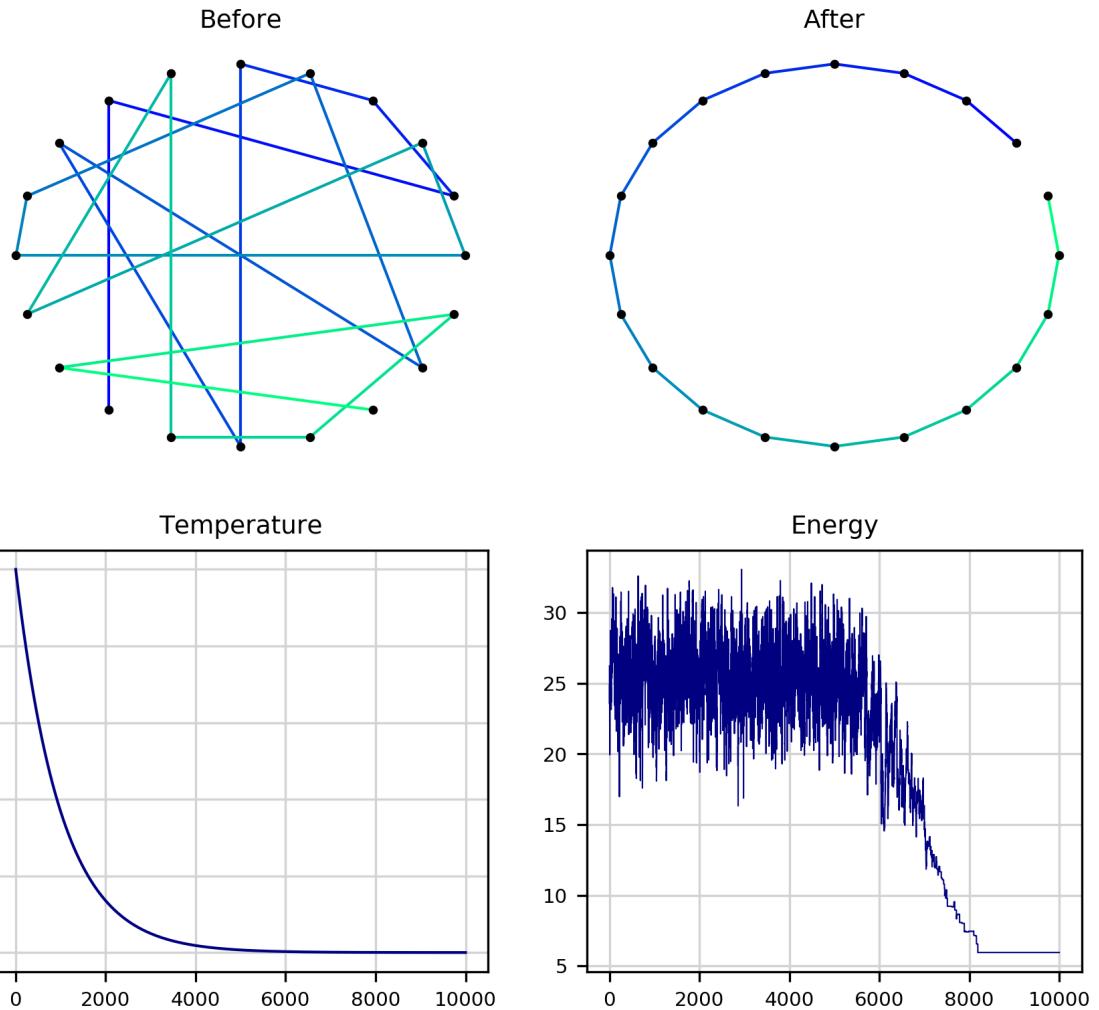
Rys. 4. $n = 100$, rozkład jednostajny, *arbitrary swap*, $T(n) = (0.999992)^n$, $I_{\max} = 2 \cdot 10^6$



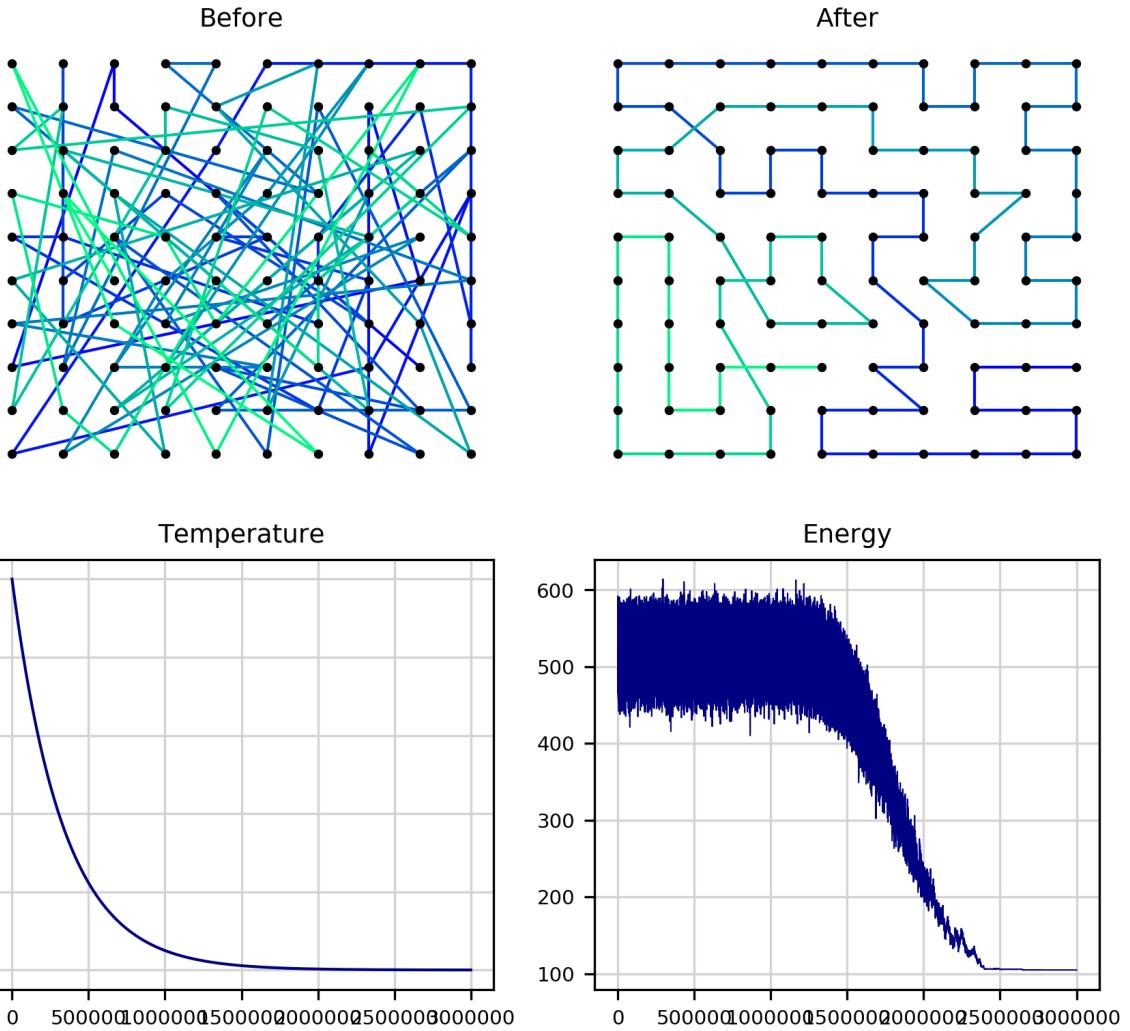
Rys. 5. $n = 20$, 4 grupy, arbitrary swap, $T(n) = (0.99998)^n$, $I_{\max} = 7 \cdot 10^5$



Rys. 6. $n = 50$, 4 grupy, arbitrary swap, $T(n) = (0.99999)^n$, $I_{\max} = 2 \cdot 10^6$



Rys. 7. $n = 20$, arbitrary swap, $T(n) = (0.999)^n$, $I_{\max} = 1 \cdot 10^4$



Rys. 8. $n = 10 \times 10$, arbitrary swap, $T(n) = (0.999997)^n$, $I_{\max} = 3 \cdot 10^6$

Zadanie 2. Obraz binarny

Energia bitmapy jest sumą energii każdego czarnego punktu na bitmapie

$$E(B) = \sum_{P \in B} E(P), \quad (4)$$

natomast energia punktu bitmapy jest iloczynem skalarnym wektora bitowego obecności czarnych sąsiadów i wektora współczynników sąsiedztwa o wartościach rzeczywistych

$$E(P) = \vec{N} \cdot \vec{\mu} = \sum_i 1_{B[P_i]} \mu_i \quad (5)$$

Zostały założone periodyczne warunki brzegowe, tzn.

$$B[x][y] = B[x + aw][y + bh], \quad a, b \in \mathbb{Z}, \quad (6)$$

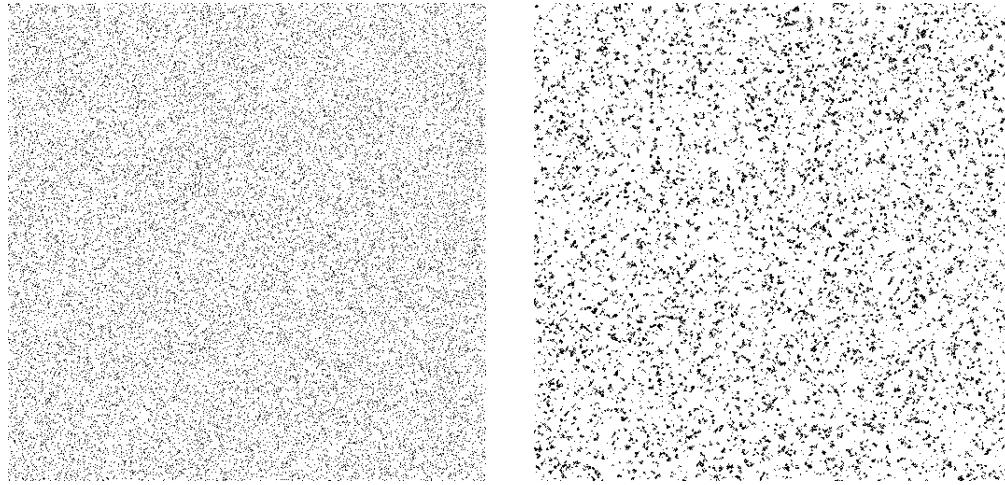
gdzie w, h oznaczają odpowiednio szerokość i wysokość bitmapy 2D.

Sąsiedztwo jest zdefiniowane jako mapa, w której kluczami są współrzędne względem rozpatrywanego punktu, a wartościami są współczynniki μ .

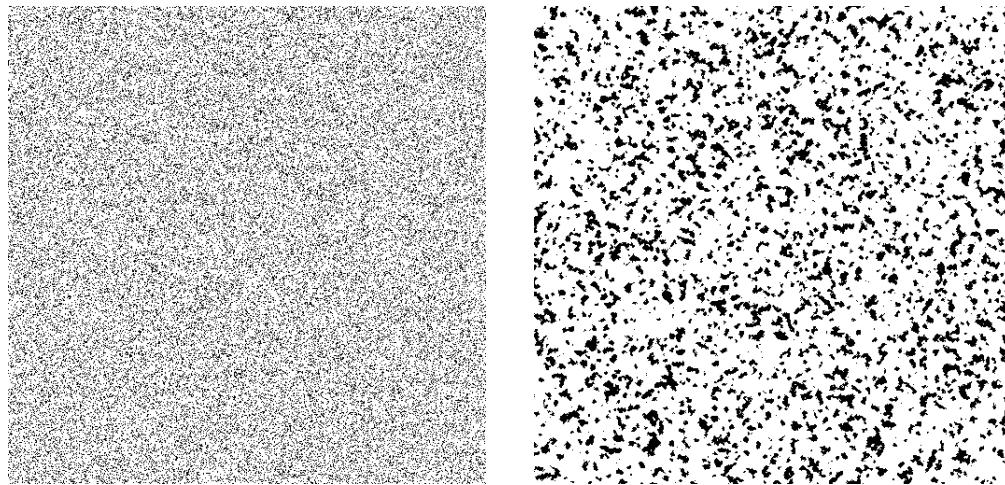
```
neighbourhood = {
    (-2,-2): +1, (-1,-2): +1, (0,-2): +1, (1,-2): +1, (2,-2): +1,
    (-2,-1): +1, (-1,-1): -1, (0,-1): -1, (1,-1): -1, (2,-1): +1,
    (-2, 0): +1, (-1, 0): -1, (0, 0): -1, (1, 0): -1, (2, 0): +1,
    (-2, 1): +1, (-1, 1): -1, (0, 1): -1, (1, 1): -1, (2, 1): +1,
    (-2, 2): +1, (-1, 2): +1, (0, 2): +1, (1, 2): +1, (2, 2): +1,
} # negative 8-neighbourhood with positive 16-neighbourhood
```

Aby zmniejszyć złożoność algorytmu oraz przyspieszyć działanie programu, zastosowany został szereg optymalizacji pozwalający uniknąć konieczności obliczania energii układu w każdej iteracji symulowanego wyżarzania. W przypadku przenoszenia jednego czarnego punktu w inne miejsce bitmapy (*arbitrary swap*) na zmianę energii układu wpływają jedynie zmiany energii obszarów sąsiedztwa dookoła modyfikowanych punktów bitmapy.

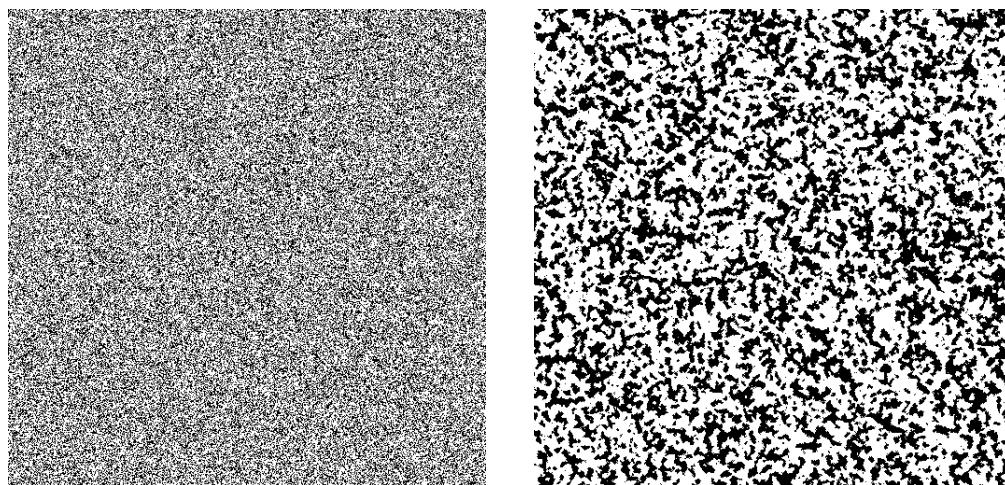
```
def calculate_region_energy(bitmap, neighbourhood, point):
    x, y = point
    if bitmap[x][y] == 0:
        return 0
    width, height = size(bitmap)
    return sum([
        weight if bitmap[(x-dx)%width][(y-dy)%height] else 0
        for (dx, dy), weight in neighbourhood.items()
    ]) + sum([
        weight if bitmap[(x+dx)%width][(y+dy)%height] else 0
        for (dx, dy), weight in neighbourhood.items()
    ])
```



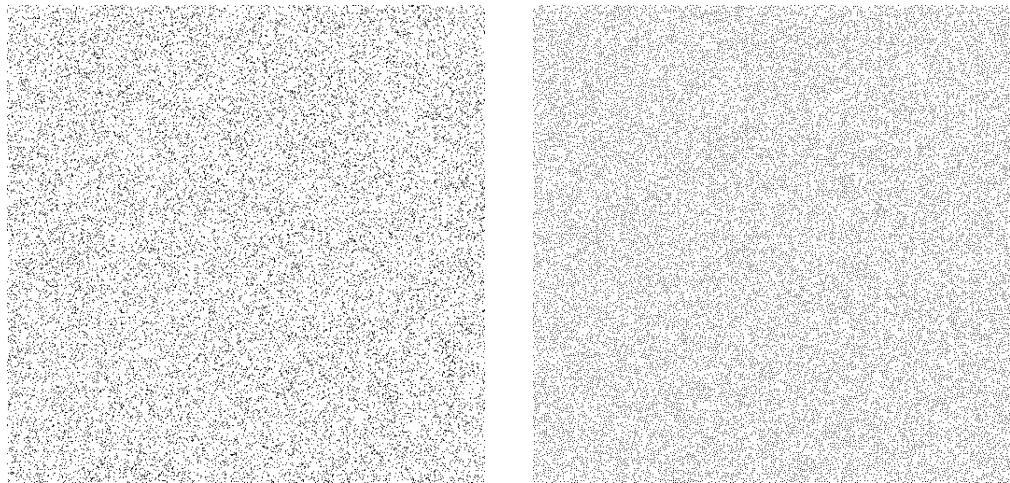
Rys. 9. 8-sąsiedztwo obniżające energię, $\delta = 0.1$



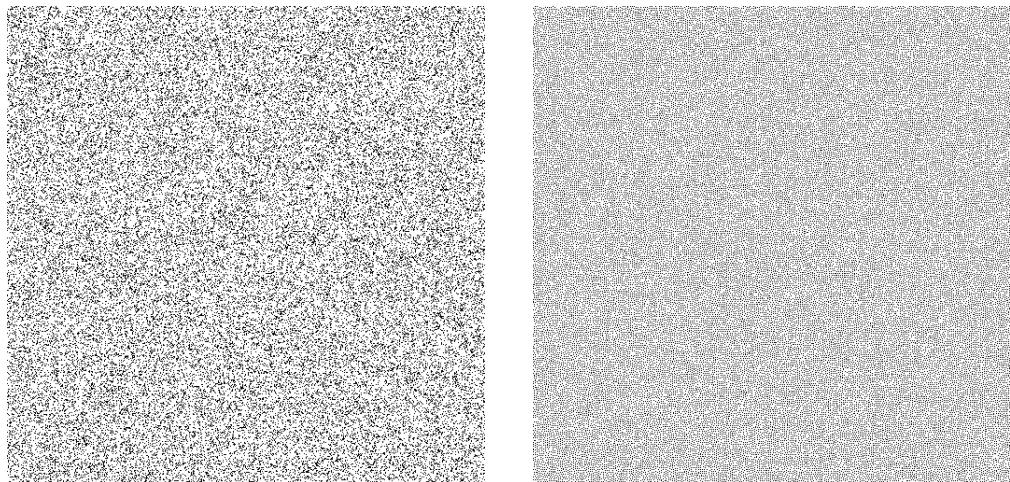
Rys. 10. 8-sąsiedztwo obniżające energię, $\delta = 0.2$



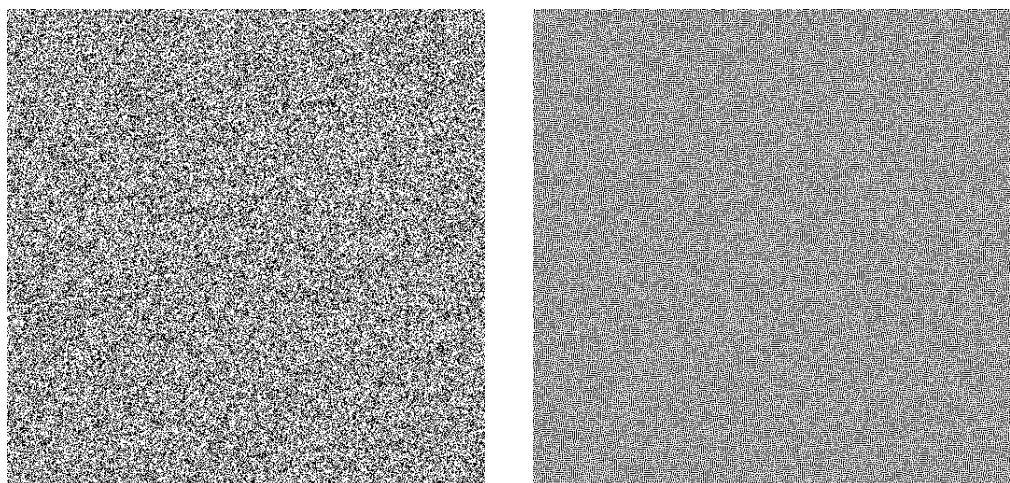
Rys. 11. 8-sąsiedztwo obniżające energię, $\delta = 0.4$



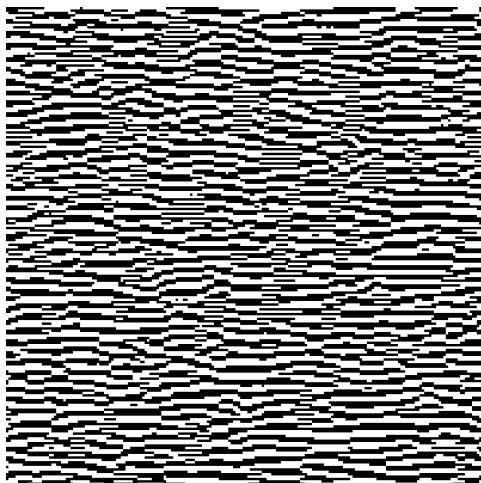
Rys. 12. 8-sąsiedztwo podwyższające energię, $\delta = 0.1$



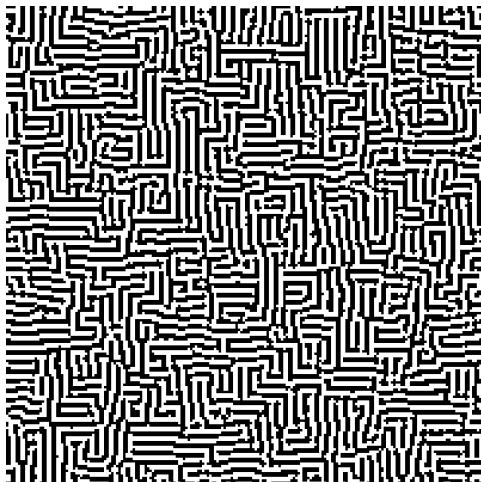
Rys. 13. 8-sąsiedztwo podwyższające energię, $\delta = 0.2$



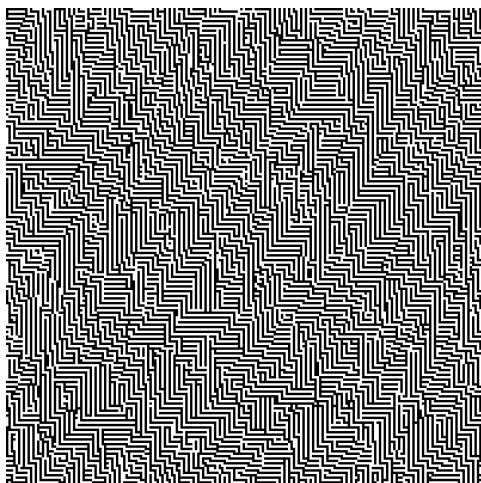
Rys. 14. 8-sąsiedztwo podwyższające energię, $\delta = 0.4$



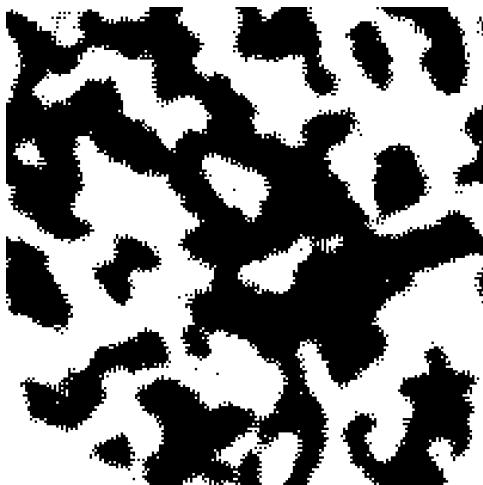
Rys. 15. Sąsiedztwo horyzontalne obniżające + wertykalne podwyższające energię, $\delta = 0.5$



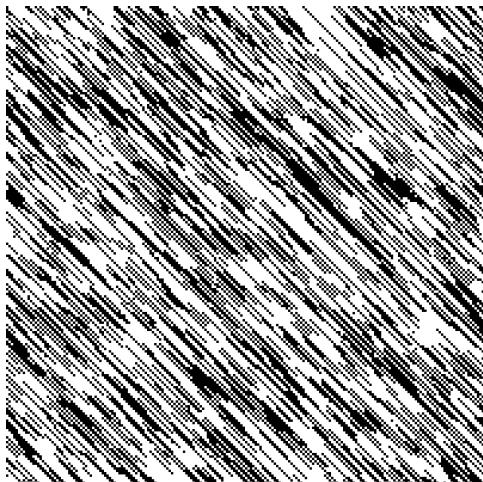
Rys. 16. 8-sąsiedztwo obniżające energię + 16-sąsiedztwo podwyższające energię, $\delta = 0.5$



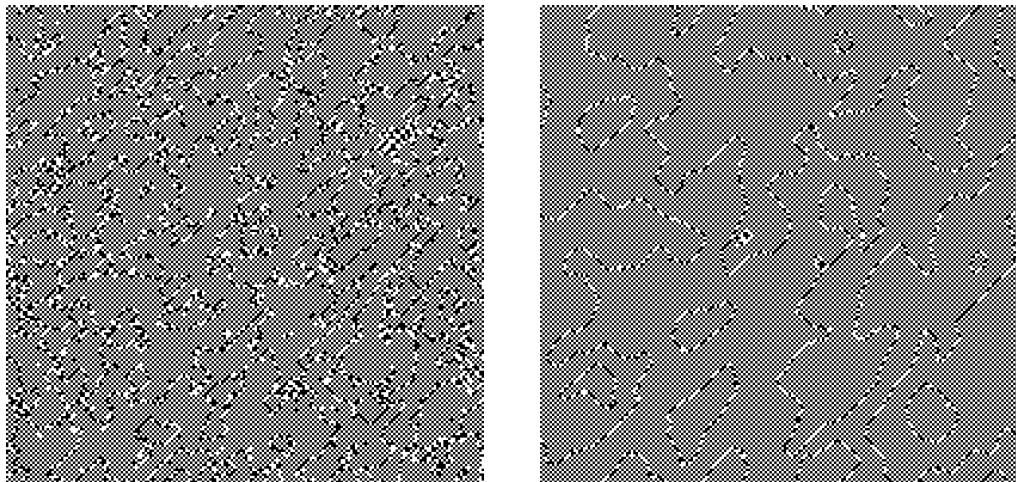
Rys. 17. Sąsiedztwo w kształcie litery S obniżające energię, $\delta = 0.5$



Rys. 18. 16-sąsiedztwo obniżające energię, $\delta = 0.5$



Rys. 19. Sąsiedztwo diagonalne, $\delta = 0.5$



Rys. 20. Sąsiedztwo w kształcie litery X obniżające energię. Porównanie wynikowej bitmapy dla różnych funkcji temperatury: (a) $T(n) = (0.9999)^n$, (b) $T(n) = (0.99998)^n$. Bitmapa po prawej stronie jest lepiej wyżarzona niż ta po lewej stronie z uwagi na dłuższy czas chłodzenia.

Zadanie 3. Sudoku

Każdy kwadrat 3×3 jest uzupełniany brakującymi cyframi (w losowej kolejności). Zapewnia to właściwą liczbę wystąpień każdej cyfry w całym sudoku.

Kolejny stan jest generowany przez zamianę dowolnych dwóch cyfr w obrębie jednego kwadratu 3×3 , które były puste na początku rozwiązywania zagadki.

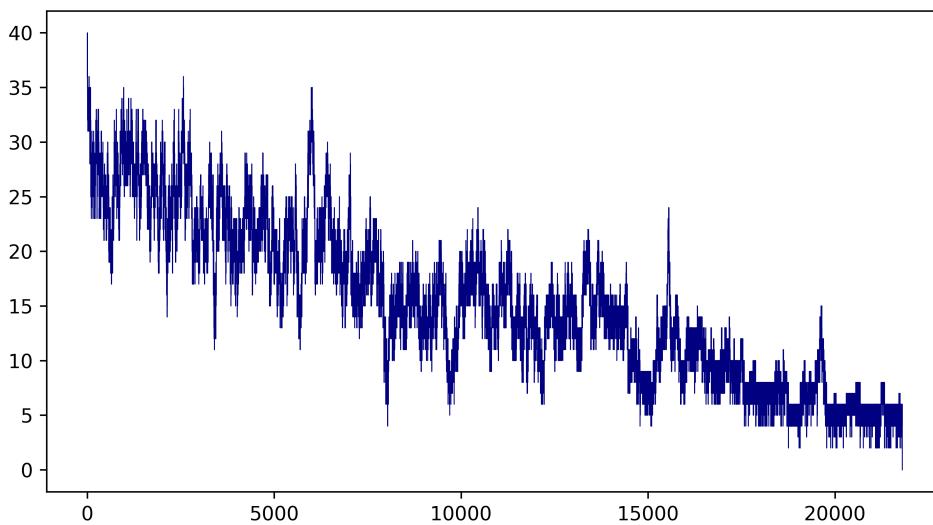
Funkcją energii układu jest liczba powtórzeń każdej cyfry w każdym wierszu i w każdej kolumnie planszy. Energia układu jest zatem liczbą naturalną z przedziału $[0, 162]$. Jeśli energia układu jest równa zero, oznacza to, że sudoku zostało rozwiązane.

Należy zwrócić uwagę, że algorytm symulowanego wyżarzania służy do minimalizacji funkcji, ale nie gwarantuje znalezienia minimum globalnego. W przypadku trudnych sudoku (np. zawierających dokładnie 17 uzupełnionych pól) istnieje wiele minimów lokalnych (o wartości 2, 3, 4), co wynika ze specyfiki problemu rozwiązywania sudoku. W pozostałych przypadkach łatwych oraz średnio-trudnych sudoku program znajduje rozwiązanie zagadki w czasie rzędu 1 s.

Plansze testowe zostały zaczerpnięte ze strony https://projecteuler.net/project/resources/p096_sudoku.txt.

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

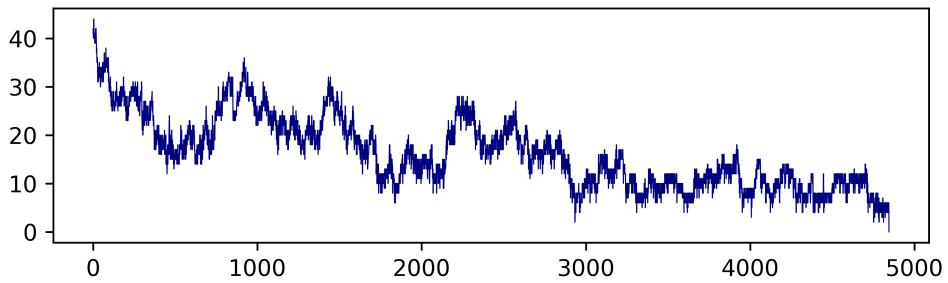
4	8	3	9	2	1	6	5	7
9	6	7	3	4	5	8	2	1
2	5	1	8	7	6	4	9	3
5	4	8	1	3	2	9	7	6
7	2	9	5	6	4	1	3	8
1	3	6	7	9	8	2	4	5
3	7	2	6	8	9	5	1	4
8	1	4	2	5	3	7	6	9
6	9	5	4	1	7	3	8	2



Rys. 21. Grid 1

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

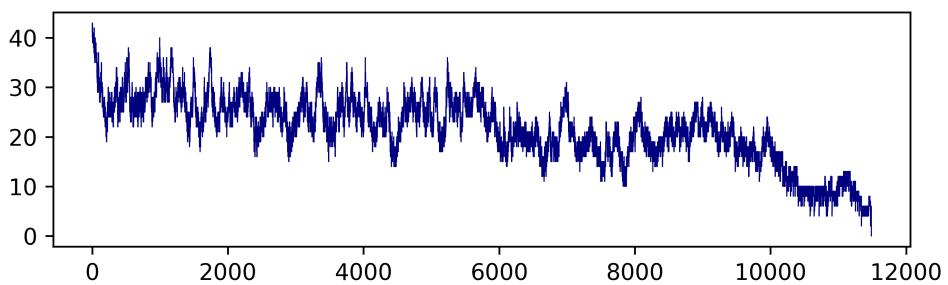
5	2	3	8	1	6	7	4	9
7	8	4	5	9	3	1	2	6
6	9	1	4	7	2	8	3	5
2	3	9	1	4	5	6	8	7
4	5	7	2	6	8	9	1	3
1	6	8	9	3	7	2	5	4
3	4	2	7	8	9	5	6	1
9	1	5	6	2	4	3	7	8
8	7	6	3	5	1	4	9	2



Rys. 22. Grid 5

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

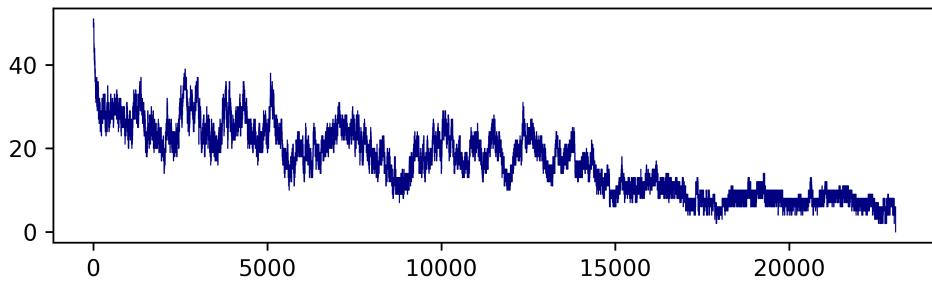
7	6	1	9	2	8	4	5	3
9	2	5	7	4	3	1	6	8
4	3	8	6	1	5	9	2	7
3	5	7	4	6	1	2	8	9
8	9	4	3	7	2	6	1	5
2	1	6	5	8	9	3	7	4
6	8	9	1	5	4	7	3	2
1	4	2	8	3	7	5	9	6
5	7	3	2	9	6	8	4	1



Rys. 23. Grid 10

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

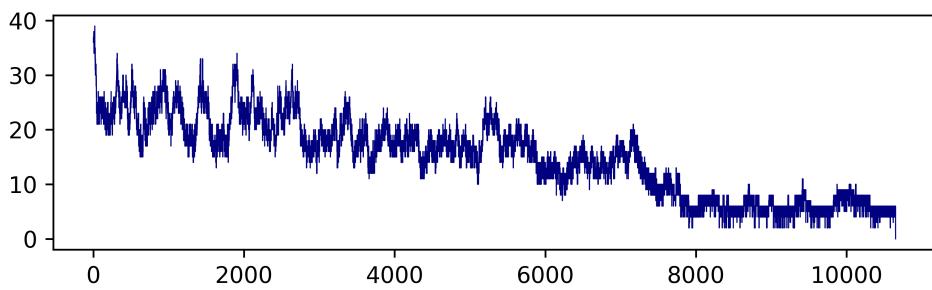
6	9	7	1	2	8	3	4	5
4	2	8	6	3	5	1	9	7
3	1	5	4	7	9	6	8	2
5	3	1	2	4	6	9	7	8
2	8	6	3	9	7	4	5	1
9	7	4	5	8	1	2	6	3
1	4	9	8	5	2	7	3	6
7	5	2	9	6	3	8	1	4
8	6	3	7	1	4	5	2	9



Rys. 24. Grid 15

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

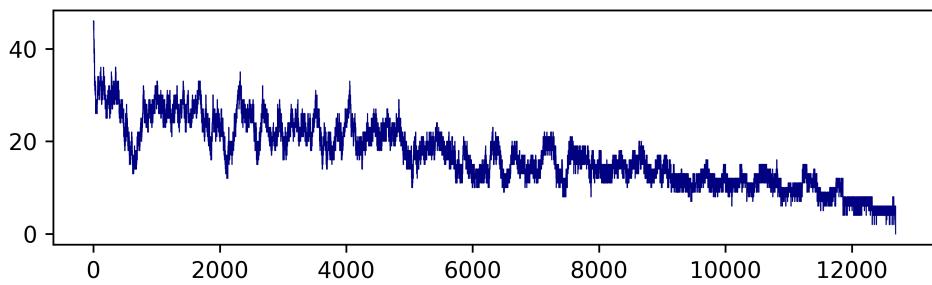
7	8	2	6	1	4	3	5	9
4	3	9	8	2	5	1	7	6
6	5	1	9	3	7	4	2	8
2	9	3	4	7	1	8	6	5
5	6	8	3	9	2	7	1	4
1	4	7	5	6	8	2	9	3
3	2	6	7	4	9	5	8	1
9	7	5	1	8	3	6	4	2
8	1	4	2	5	6	9	3	7



Rys. 25. Grid 20

3	6		2		8	9
			3	6	1	
8	3			6		2
4		6	3			7
6	7			1		8
		4	1	8		
9	7		3		1	4

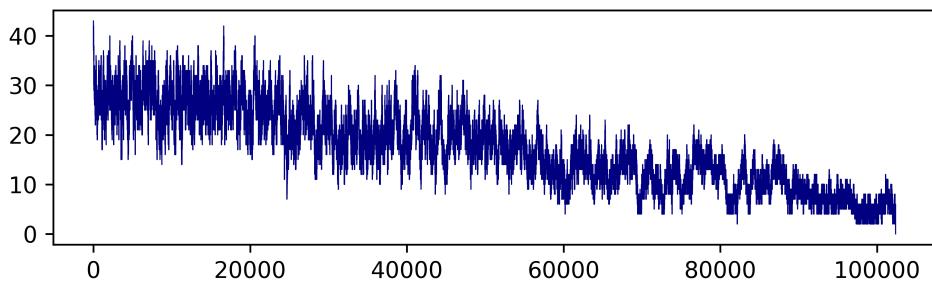
3	6	1	5	2	4	7	8	9
7	8	9	3	6	1	4	2	5
5	2	4	8	7	9	3	6	1
8	9	3	1	5	7	6	4	2
4	1	2	6	8	3	5	9	7
6	5	7	9	4	2	1	3	8
1	4	8	7	9	6	2	5	3
2	3	5	4	1	8	9	7	6
9	7	6	2	3	5	8	1	4



Rys. 26. Grid 25

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

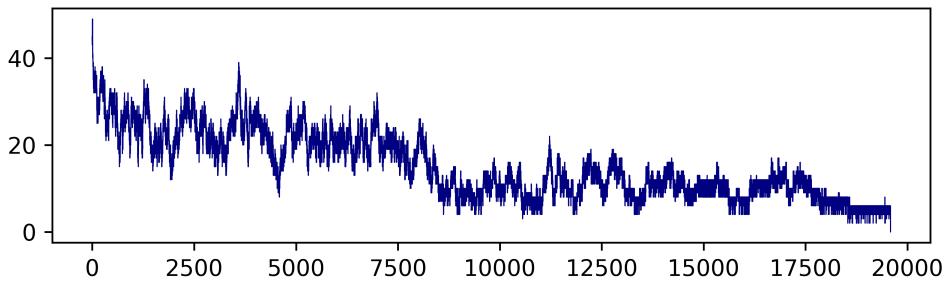
2	9	8	1	7	5	6	4	3
6	5	7	3	9	4	1	2	8
1	3	4	2	8	6	5	7	9
8	2	1	6	4	9	7	3	5
5	7	3	8	2	1	4	9	6
4	6	9	7	5	3	2	8	1
3	1	2	4	6	8	9	5	7
7	8	5	9	1	2	3	6	4
9	4	6	5	3	7	8	1	2



Rys. 27. Grid 30

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

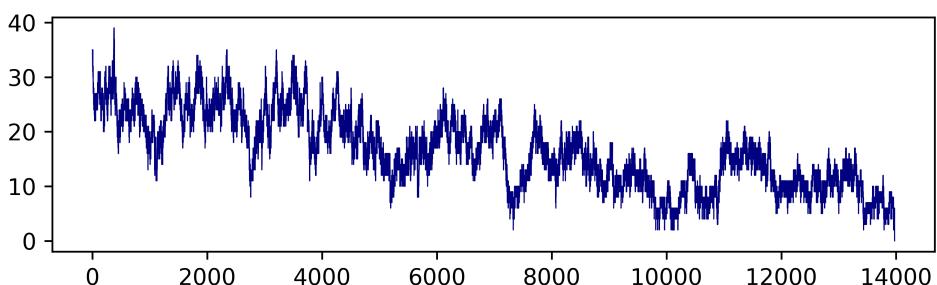
4	5	3	2	1	8	7	9	6
6	2	9	7	5	3	4	8	1
1	7	8	4	9	6	5	3	2
7	9	6	5	8	2	3	1	4
3	1	4	9	6	7	8	2	5
2	8	5	1	3	4	6	7	9
5	4	2	8	7	9	1	6	3
9	3	7	6	4	1	2	5	8
8	6	1	3	2	5	9	4	7



Rys. 28. Grid 35

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

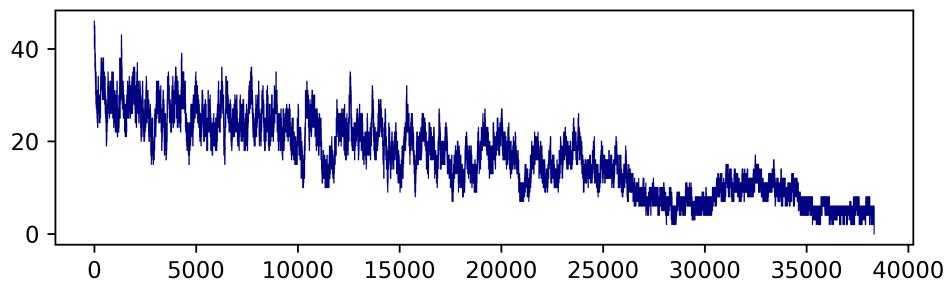
1	9	3	6	7	2	4	8	5
4	6	2	3	5	8	9	7	1
7	8	5	9	1	4	6	2	3
5	3	8	2	9	6	7	1	4
6	7	4	1	3	5	2	9	8
2	1	9	4	8	7	3	5	6
8	2	6	7	4	1	5	3	9
9	4	1	5	2	3	8	6	7
3	5	7	8	6	9	1	4	2



Rys. 29. Grid 40

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

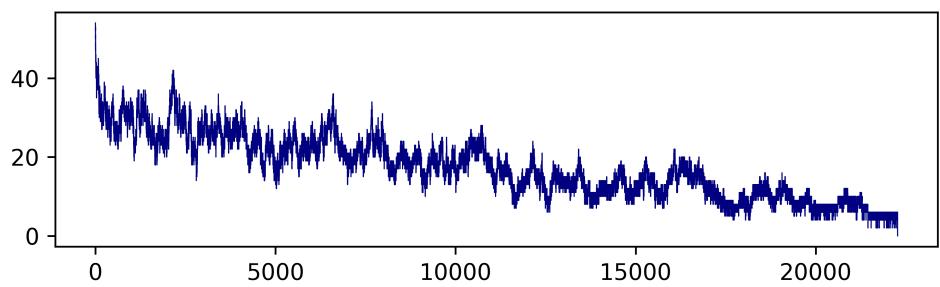
5	8	6	1	2	7	9	4	3
7	2	3	4	6	9	8	5	1
4	9	1	8	5	3	2	6	7
1	3	5	9	7	4	6	2	8
2	7	9	6	1	8	5	3	4
6	4	8	5	3	2	1	7	9
9	1	7	2	4	6	3	8	5
3	5	2	7	8	1	4	9	6
8	6	4	3	9	5	7	1	2



Rys. 30. Grid 45

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

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



Rys. 31. Grid 50

Bibliografia

- [1] https://www.icsr.agh.edu.pl/~mownit/pdf/18_simulated_annealing_v2.pdf
- [2] <https://www.mimuw.edu.pl/~grygiel/woen/woen4.pdf>
- [3] https://en.wikipedia.org/wiki/Simulated_annealing
- [4] <https://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny>