Projektowanie Efektywnych Algorytmów

Projekt

23.12.2022

259062 Jacek Myjkowski

(4) Symulowane Wyżarzanie

Spis treści

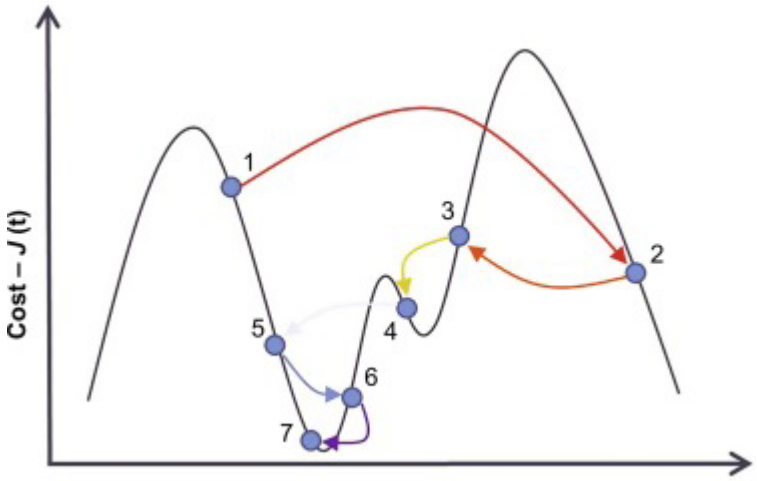
1. Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności algorytmu symulowanego wyżarzania rozwiązującego problem komiwojażera w wersji optymalizacyjnej. Problem komiwojażera, lub jego uogólnienie w postaci problemu chińskiego listonosza to przykład problemu z klasy NP trudnych. Polega on na wyznaczeniu jak najlepszej trasy zawierającej w sobie wszystkie punkty (wierzchołki grafu) przechodzącej przez każdy punkt tylko jeden raz oraz zaczynającej i kończącej się w tym samym punkcie. Wykonywane to będzie metodą Symulowanego wyżarzania będącego algorytmem opierającym się na przeszukiwaniach lokalnych. Do tej pory przetestowane przeze mnie metody to BruteForce i BranchAndBound. Spodziewam się, że metoda SA, będzie dużo szybsza od B&B, ale kosztem jakości – nie zawsze będzie gwarantowała znalezienie rozwiązania optymalnego. Także nie powinna być szybsza od razu. Dla mniejszych instancji będzie dużo wolniejsza od BF czy BB, ale za to będzie potrafiła obsłużyć wielokrotnie większe instancje. Dodatkowo będzie w dużym stopniu zależna od „strojenia" – być może nawet więcej niż od samej budowy.

Rysunek 1 Przykład trasy, która musi odwiedzić wszystkie miasta i być pętlą

2. Metoda

Metoda tutaj wykorzystana to tzw. **Metoda Symulowanego wyżarzania** zwana inaczej „**Simulated Annealing**” Polega ona na przeszukiwaniu sąsiedztwa znalezionego aktualnie rozwiązania w celu znalezienia lepszego lub akceptowalnie gorszego (po to aby w miarę możliwości nie utknąć jednym miejscu). Używamy w tym algorytmie pewnej formy losowości, ponieważ losowo najpierw generujemy rozwiązanie wyjściowe, a następnie losowo przeglądamy sąsiedztwa (sąsiednie kombinacje ścieżki w przypadku TSP, różniące się pozycją co najmniej dwóch elementów). Aby nie była to metoda całkowicie losowa używamy takiego parametru jak temperatura, która im wyższa, tym na odważniejsze (będące gorszym od aktualnego) rozwiązania pozwala się przenosić, sposobu jej obniżania oraz parametru definiującego długość epoki, czyli to ile razy przeszukamy sąsiedztwo zanim obniżymy temperaturę.



Rysunek 2 Zobrazowanie pewnego rodzaju losowości metody SA

3. Algorytm

Algorytm tutaj zaimplementowany to ogólno pojęte Symulowane Wyżarzanie z zaimplementowanym mechanizmem wielostartu, pozwalającym na wielokrotne wykonanie „pomniejszych" symulowanych wyżarzeń, które to zwiększają szanse na znalezienie optymalnego wyniku poprzez niwelowanie ryzyka utknięcia w optimum lokalnym, nie będącym jednocześnie optimum globalnym.

W tym algorytmie zaimplementowana jest również możliwość wyboru warunku stopu (stop po upłynięciu danej ilości czasu podanej w sekundach lub stop po odnalezieniu rozwiązania będącego co gorszym o maksymalnie x% od optymalnego dla tego zestawu danych).

Oprócz możliwości wyboru warunku stopu są również zaimplementowane różne mechaniki generowania sąsiedztwa, takie jak:

- dwu-zamiana – polegająca na wylosowaniu dwóch, różnych od siebie indeksów, a następnie zamienieniu ze sobą elementów znajdujących się w aktualnej tymczasowej ścieżce pod tymi indeksami

- trzy-zamiana – polega na tym samym co dwu-zamiana, z tą różnicą, że tutaj element spod wylosowanego indeksu A idzie do C, z B do A, a z C do B

- wymiana łuków – polega na tym, że odwracamy kolejnością elementy znajdujące się w naszej ścieżce pomiędzy wylosowanymi indeksami

Kolejną opcją wyboru jest wybór sposobu generowania temperatury początkowej:

- na podstawie średniego kosztu permutacji ścieżki metodą dwu-zamiany:

$$startTemperature = \frac{-1 * |średniKosztPermutacji|}{\log(alphaCoolRate)}$$

- na podstawie ilości wierzchołków w grafie oraz najdłuższej krawędzi

$$startTemperature = najdłuższaKrawędź^2 * ilośćWierzchołków^2$$

Ostatnią opcją wyboru różnych algorytmów jest możliwość wyboru schematu chłodzenia spośród trzech:

- schematu geometrycznego

$$newTemperature = oldTemperature * alphaCoolRate^k$$

Gdzie **k** to numer epoki

- schematu Boltzmann'a

$$newTemperature = \frac{-1 * oldTemperature}{alphaCoolRate + coolingB * log(k)}$$

Gdzie **k** to numer epoki, a **coolingB** to dodatkowy współczynnik

- schematu Cauchy'ego

$$newTemperature = \frac{oldTemperature}{alphaCoolRate + coolingB * k}$$

Gdzie **k** to numer epoki, a **coolingB** to dodatkowy współczynnik

Poza zmiennymi w postaci całych algorytmów istnieją zmienne w postaci parametrów liczbowych, takie jak:

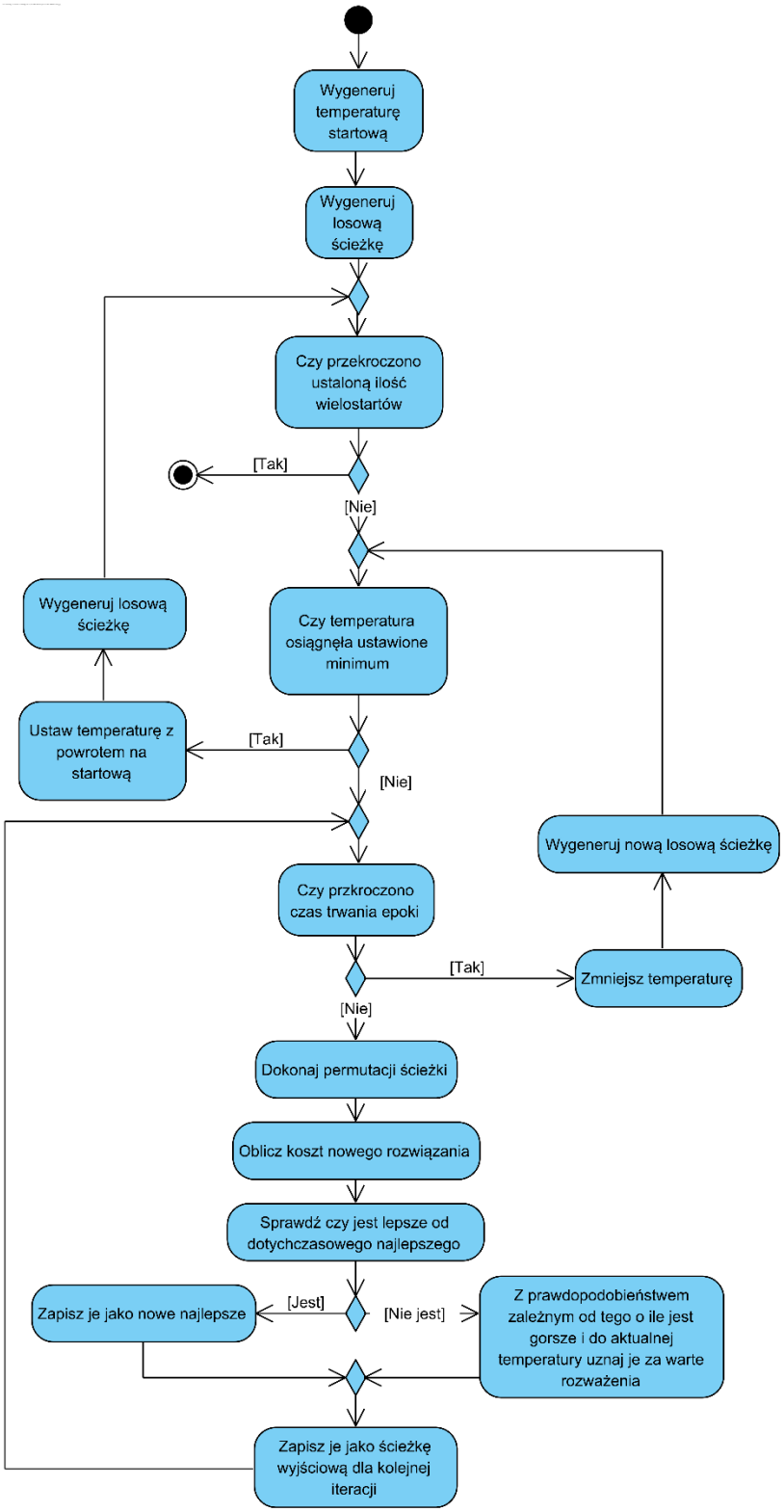
- „mistakePercents" – w nim ustawia się o ile procent znalezione rozwiązanie może być gorsze od optymalnego, aby można było uznać, że algorytm znalazł rozwiązanie

- „maxTime" – maksymalny czas, jaki algorytm ma na znalezienie jak najlepszego rozwiązania

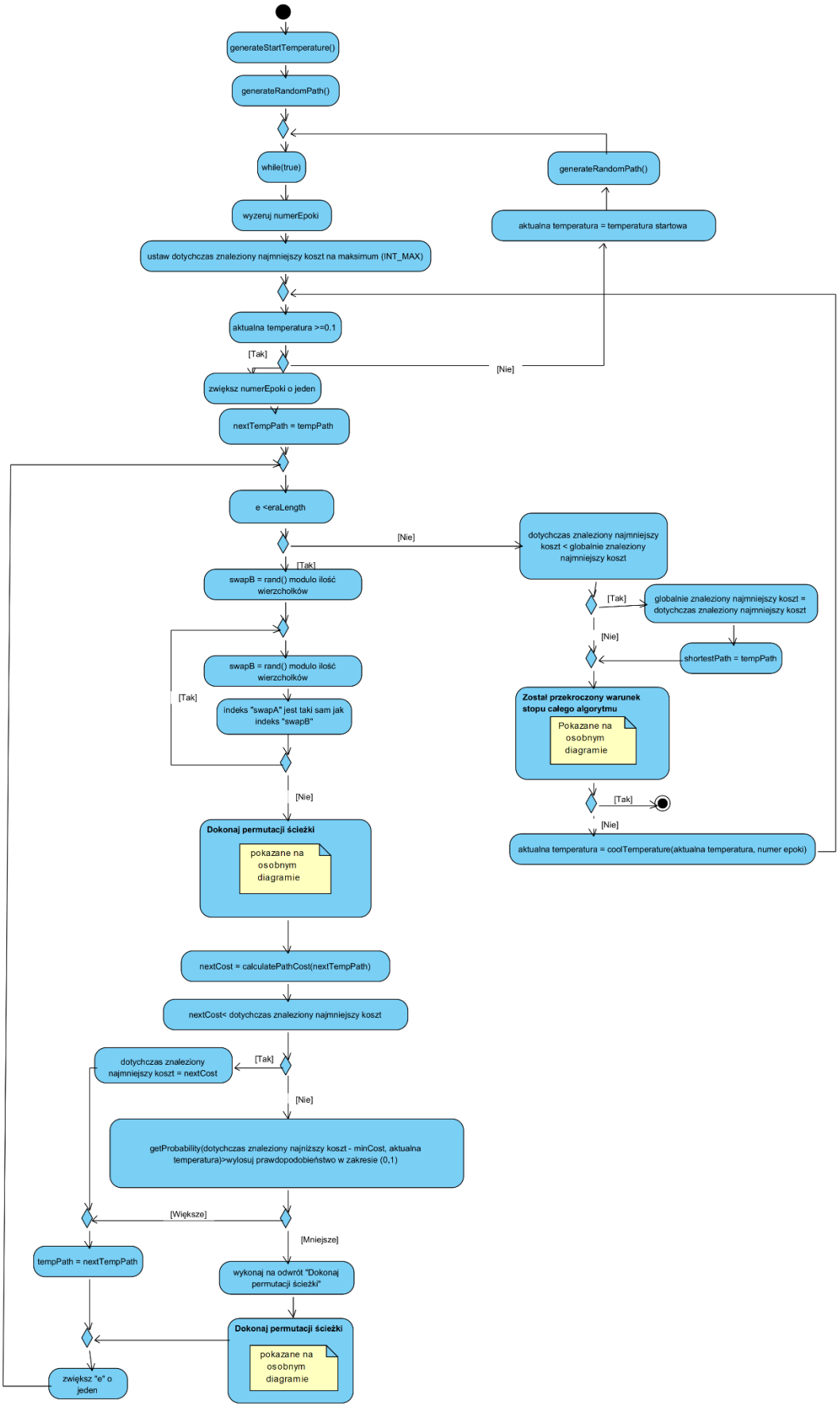
- „erasOutDivider" – po jakiej części długości ery bez żadnych zmian w ścieżce algorytm ma przejść do następnej ery

- „alphaCoolRate" – prędkość z jaką będzie opadała temperatura – im większy tym wolniej temperatura opada

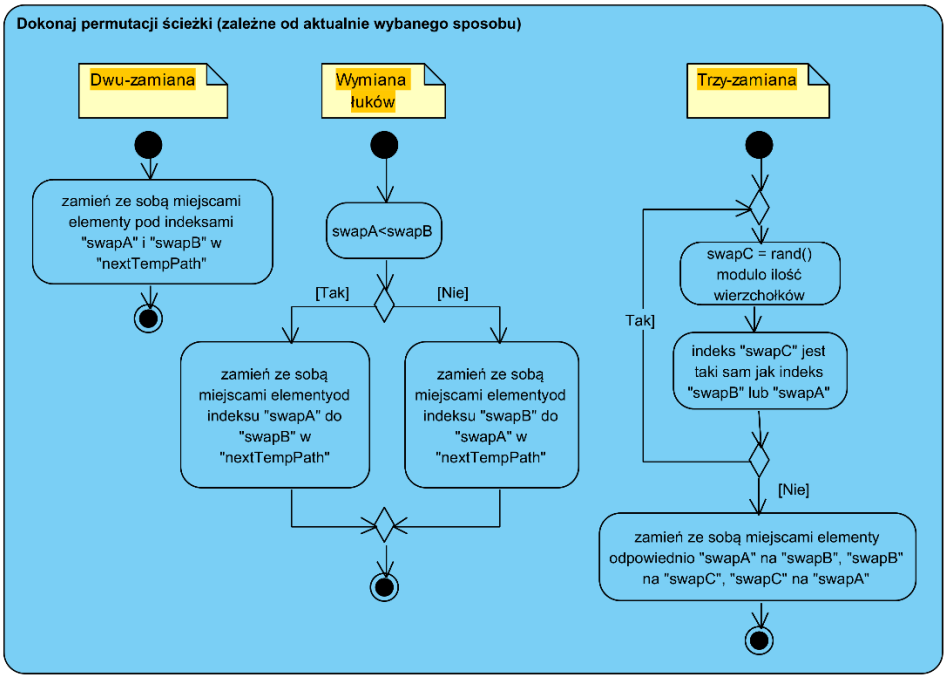
- „coolinB" – dodatkowy parametr do chłodzenia Cauchy'ego i Boltzmann'a



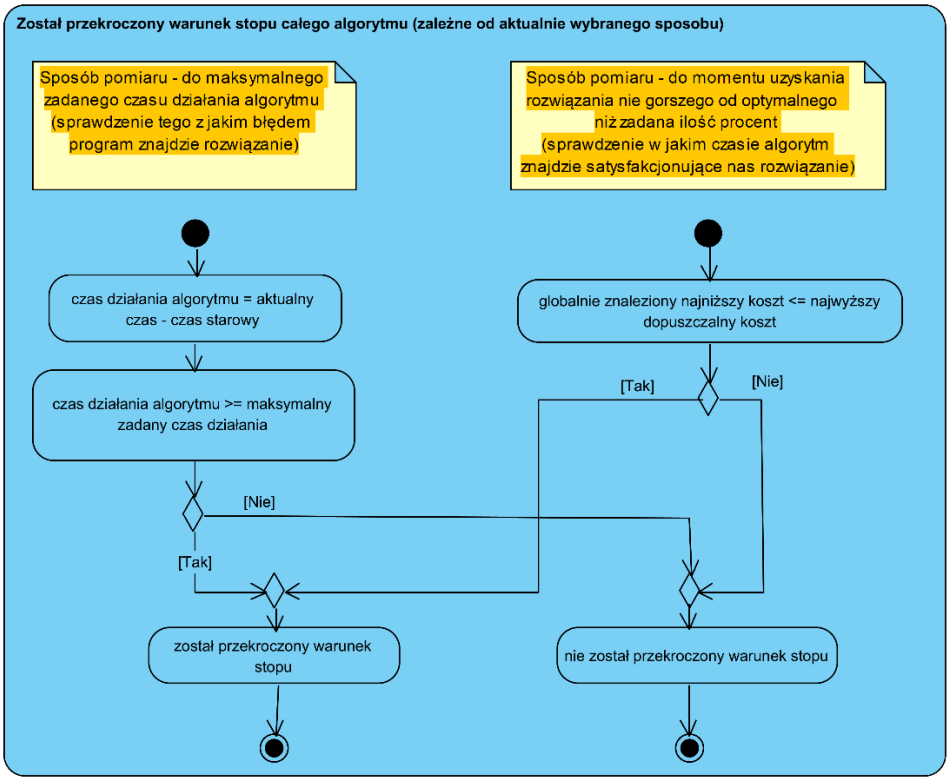
Rysunek 3 Przebieg logiczny algorytmu Simulated Annealing



Rysunek 4Uproszczony przebieg fizyczny algorytmu SA wraz z zaimplementowanym mechanizmem wielostartu



Rysunek 5Diagram przedstawiający mechanizm permutacji sąsiedztwa



Rysunek 6 Diagram przedstawiający mechanizm dostarczający możliwość opuszczenia algorytmu

4. Dane testowe

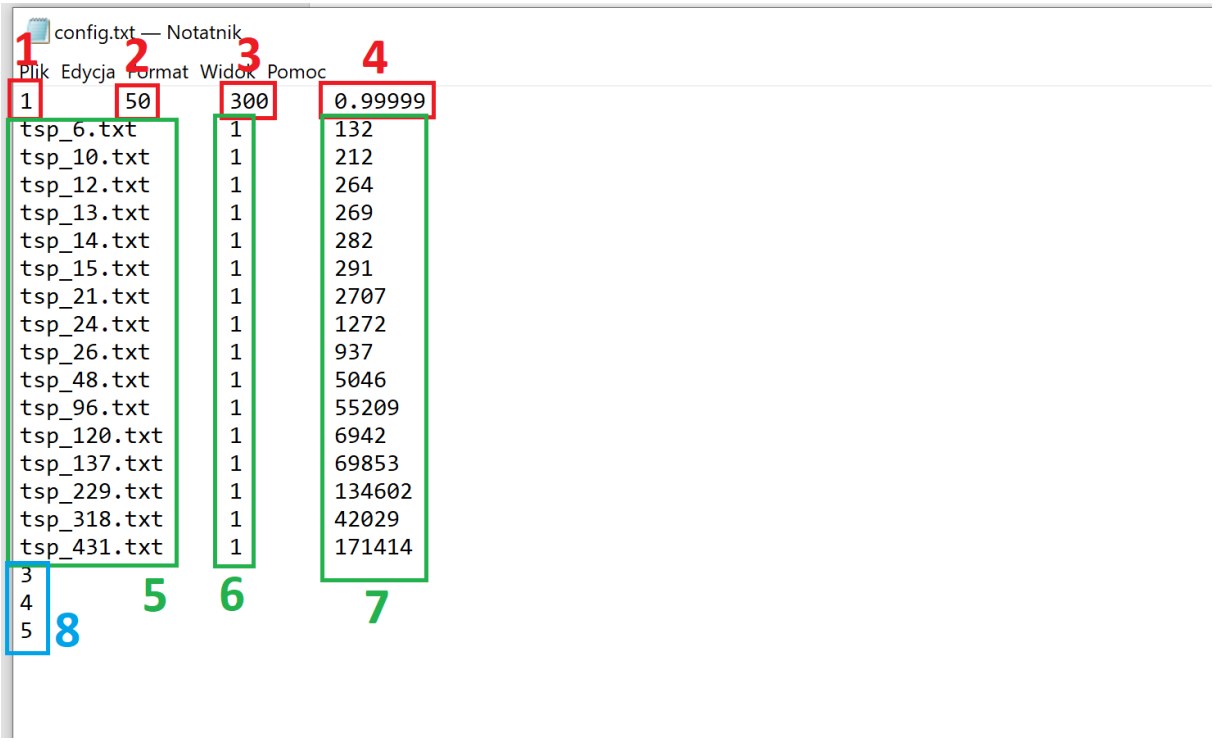
Do sprawdzenia poprawności działania algorytmu oraz do dostrojenia, wybrano następujący zestaw instancji:

- tsp_6.txt Optymalna długość ścieżki - 132
- tsp_10.txt Optymalna długość ścieżki - 212
- tsp_12.txt Optymalna długość ścieżki - 263
- tsp_13.txt Optymalna długość ścieżki - 269
- tsp_14.txt Optymalna długość ścieżki - 282
- tsp_15.txt Optymalna długość ścieżki - 291
- tsp_21.txt Optymalna długość ścieżki - 2707
- tsp_24.txt Optymalna długość ścieżki - 1272
- tsp_26.txt Optymalna długość ścieżki - 937
- tsp_48.txt Optymalna długość ścieżki - 5046
- tsp_96.txt Optymalna długość ścieżki - 55209
- tsp_120.txt Optymalna długość ścieżki - 6942
- tsp_137.txt Optymalna długość ścieżki - 69853
- tsp_229.txt Optymalna długość ścieżki - 134602
- tsp_318.txt Optymalna długość ścieżki - 42029
- tsp_431.txt Optymalna długość ścieżki - 171414

Dla których koszty optymalnej ścieżki zostały wprowadzone do pliku config.txt

5. Procedura badawcza

Zadanie polegało na zbadaniu zależności czasu rozwiązania problemu od wielkości instancji. W przypadku Symulowanego Wyżarzania jest wiele parametrów, które odpowiadają za skuteczność algorytmu, gdyż jest to algorytm niedokładny (czyli nie mamy pewności, że zawsze da wynik optymalny). Program jest sterowany plikiem „config.txt”:



Rysunek 7 Prezentacja pliku konfiguracyjnego wraz z zaznaczonymi jego sekcjami

1. Wybór trybu mierzenia:
   a. 0 – do określonego maksymalnego czasu
   b. 1 – do uzyskania wyniku będącego maksymalnie o x% gorszym od optymalnego
2. Procent jakości – określenie o ile procent nasz wynik może być gorszy od optymalnego
3. Maksymalny czas [s] (działa tylko z trybem mierzenia „0”) – określa po upływie jakiego czasu nasz algorytm się zakończy
4. Współczynnik chłodzenia „alpha” – określa jak wolno ma się zmniejszać temperatura (im więcej 9 po przecinku tym wolniejsze ochładzanie)
5. Nazwy plików z danymi
6. Ilość powtórzeń wykonania całego badania dla każdego z plików
7. Wprowadzenie ścieżki optymalnej dla danej instancji
8. 3 wartości parametru długości epoki, obliczanego wzorem:

$$długośćEpoki = liczbaWierzchołków^{parametrDługościEpoki}$$

Każdy z parametrów testowałem na jak największej liczbie instancji za pomocą pętli, aby jak najbardziej zminimalizować różnice w dostępie do procesora przez program.

Do pliku wyjściowego „daneWynikowe_xxx.txt” wypisywane są czasy wykonania każdego algorytmu, oraz optymalna ścieżka wraz z jej kosztem i tym, o ile rozwiązanie jest gorsze

od optymalnego porównawczego oraz niezbędne parametry każdego wywołanego algorytmu. Następnie wyniki zostały obrobione w programie MS Excel.

6. Wyniki

Wyniki zostały zgormadzone w plikach „daneWynikowe_xxx.txt” (gdzie „xxx” to krótki opis, co w tym pliku było badane, np.”300sek_porównaniePrzegladuSasiedztwa” oznacza, że były to pomiary ustawione na maksymalny czas – 300 sekund, mające na celu porównanie działania różnych mechanizmów przeglądu sąsiedztwa)

SA porównanie chłodzenia[ile procent gorsze od optymalnego rozwiązania]

Wielkość instancji	Geometryczne	Boltzmanna	Cauchyego
15	2,75%	26,46%	12,37%
21	14,44%	41,41%	28,19%
24	8,81%	54,01%	6,37%
26	24,01%	56,35%	28,60%
48	25,96%	171,09%	24,79%
96	38,47%	391,34%	51,68%
120	127,17%	468,22%	109,84%
137	55,83%	597,00%	54,96%
229	114,46%	757,50%	101,53%
318	198,41%	1123,80%	199,14%

Tabela 1 Dane zebrane podczas porównywania efektywności różnych sposobów chłodzenia

SA porównanie sposobu generowania temperatury[ile procent gorsze od optymalnego rozwiązania]

Wielkość instancji	Na podstawie średniego kosztu permutacji ścieżki	Na podstawie ilości wierzchołków i najdłuższego połączenia
15	2,75%	28,87%
21	17,73%	19,69%
24	27,91%	27,44%
26	28,18%	18,57%
48	27,45%	29,37%
96	34,18%	58,00%
120	107,85%	135,67%
137	58,62%	54,10%
229	102,78%	120,40%

Tabela 2 Dane zebrane podczas porównywania efektywności różnych sposobów generowania temperatury początkowej

SA zbadanie wpływu parametru alpha na wynik [ile procent gorsze od optymalnego rozwiązania]

Wielkość instancji	a = 0,99	a=0,999	a=0,9999	a=0,99999
6	19,70%	19,70%	19,70%	27,27%
10	11,79%	0,47%	29,72%	17,92%
12	19,32%	19,70%	0%	26,89%
13	28,62%	17,84%	28,62%	28,62%
14	0,00%	28,72%	29,08%	29,43%
15	29,55%	21,65%	27,15%	28,87%
21	29,11%	23,27%	25,12%	18,06%
24	23,35%	22,72%	29,48%	26,10%
26	22,95%	27,11%	28,60%	27,75%
48	27,27%	24,38%	29,27%	29,90%
96	39,99%	29,03%	32,17%	42,31%
120	79,36%	56,54%	42,64%	36,72%
137	59,63%	66,72%	64,71%	66,47%
229	98,93%	71,50%	90,74%	119,35%
318	168,69%	140,72%	150,85%	143,60%
431	254,81%	185,89%	139,34%	123,60%

Tabela 3 Dane zebrane podczas testowania różnych wartości parametru alphaCoolingRate dla| maxTime=300s | Chłodzenia geometrycznego | Przeglądu sąsiedztwa: Dwu-zamiany | Zadowalającej jakości wyniku = 30% | Długość ery = 3

SA zbadanie wpływu przydzielonego czasu na rozwiązanie na wynik [ile procent gorsze od optymalnego rozwiązania]

Wielkość instancji	75,000	150,000	300,000	600,000	1200,000	2400,000
tsp_6.txt	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tsp_10.txt	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tsp_12.txt	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tsp_13.txt	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tsp_14.txt	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tsp_15.txt	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tsp_21.txt	3,92%	3,92%	3,92%	3,92%	3,55%	3,92%
tsp_24.txt	3,85%	3,85%	3,85%	3,85%	4,40%	3,85%
tsp_26.txt	3,42%	3,42%	3,42%	3,42%	3,95%	3,42%
tsp_48.txt	8,68%	6,86%	8,68%	6,42%	4,54%	6,42%
tsp_96.txt	39,76%	40,63%	41,54%	33,43%	32,45%	32,91%
tsp_120.txt	62,40%	67,17%	53,54%	39,27%	40,19%	34,60%
tsp_137.txt	73,97%	72,29%	67,13%	62,44%	61,42%	52,80%
tsp_229.txt	100,95%	92,66%	87,35%	78,43%	68,71%	76,47%
tsp_318.txt	145,23%	148,26%	131,69%	128,89%	123,62%	124,04%
tsp_431.txt	146,15%	150,02%	140,62%	140,94%	130,39%	145,70%

Tabela 4 Dane zebrane podczas testowania jakości otrzymanego wyniku w konkretnie ustalonym maksymalnym czasie dla| alphaCoolingRate=0,9999 | Chłodzenia geometrycznego | Przeglądu sąsiedztwa: Dwu-zamiany | Zadowalającej jakości wyniku = 5% | Długość ery = 3

SA zbadanie wpływu parametru długości ery na wynik [ile procent gorsze od optymalnego rozwiązania]

Wielkość instancji	eraLength = 2	eraLength = 3	eraLength = 4
6	0,00%	0,00%	0,00%
10	0,00%	0,00%	0,00%
12	0,00%	0,00%	0%
13	0,00%	0,00%	0,00%
14	0,00%	0,00%	0,00%
15	0,00%	0,00%	0,00%
21	0,00%	0,00%	0,00%
24	4,72%	3,93%	3,77%
26	4,48%	4,80%	4,80%
48	4,12%	5,87%	13,08%
96	29,91%	32,93%	27,11%
120	42,01%	45,32%	44,37%
137	50,27%	64,30%	62,06%
229	74,63%	79,41%	
318	122,01%	125,21%	
431	138,09%	140,12%	

Tabela 5 Dane zebrane podczas testowania jakości otrzymanego wyniku w pod wpływem zmieniającego się parametru długości ery dla| alphaCoolingRate=0,9999 | Chłodzenia geometrycznego | Przeglądu sąsiedztwa: Dwu-zamiany | Zadowalającej jakości wyniku = 5% | maxTime=300s

SA zbadanie wpływu parametru zakończenia ery na podstawie ilości iteracji nie przynoszących zmian [ile procent gorsze od optymalnego rozwiązania]

Wielkość instancji	brak	eraLength/3	eraLength/6	eraLength/9
6	0,00%	0,00%	0,00%	0,00%
10	0,00%	0,00%	0,00%	0,00%
12	0,00%	0,00%	0,00%	0,00%
13	0,00%	0,00%	0,00%	0,00%
14	0,00%	0,00%	0,00%	0,00%
15	4,12%	0,00%	0,00%	0,00%
21	3,99%	3,92%	0,00%	0,00%
24	3,99%	3,85%	3,54%	4,40%
26	4,91%	3,42%	4,59%	3,95%
48	6,68%	8,68%	2,50%	6,40%
96	38,99%	41,54%	30,88%	29,53%
120	59,74%	53,54%	48,91%	35,59%
137	76,69%	67,13%	60,53%	52,91%
229	88,20%	87,35%	86,05%	73,21%
318	129,92%	131,69%	131,39%	136,32%
431	144,21%	140,62%	144,44%	136,54%

Tabela 6 Dane zebrane podczas testowania jakości otrzymanego wyniku w pod wpływem zmieniającego się parametru części ery, pod rząd, nie przynoszącej żadnych zmian dla| alphaCoolingRate=0,9999 | Chłodzenia geometrycznego | Przeglądu sąsiedztwa: Dwu-zamiany | Zadowalającej jakości wyniku = 5% | maxTime=300s|Długość ery = 3

SA zbadanie wpływu sposobu generowania sąsiedztwa na wynik [ile procent gorsze od optymalnego rozwiązania]

Wielkość instancji	2-zamiana	Zamiana łuków	3-zamiana
6	0,00%	0,00%	0,00%
10	0,00%	0,00%	0,00%
12	0,00%	0,00%	0%
13	0,00%	0,00%	4,46%
14	0,00%	0,00%	4,26%
15	0,00%	0,00%	2,75%
21	3,47%	3,25%	22,79%
24	1,10%	4,25%	36,24%
26	2,03%	3,74%	44,18%
48	4,88%	4,48%	117,04%
96	28,21%	14,11%	278,01%
120	50,36%	31,46%	349,55%
137	61,95%	28,97%	431,42%
229	86,14%	54,75%	575,04%
318	122,00%	112,92%	1077,63%
431	140,66%	125,48%	1129,94%

Tabela 7 Dane zebrane podczas testowania jakości otrzymanego wyniku w pod wpływem zmieniającego się algorytmu generowania sąsiedztwa dla| alphaCoolingRate=0,9999 | Chłodzenia geometrycznego | Zadowalającej jakości wyniku = 5% | maxTime=300s|Długość ery = 3

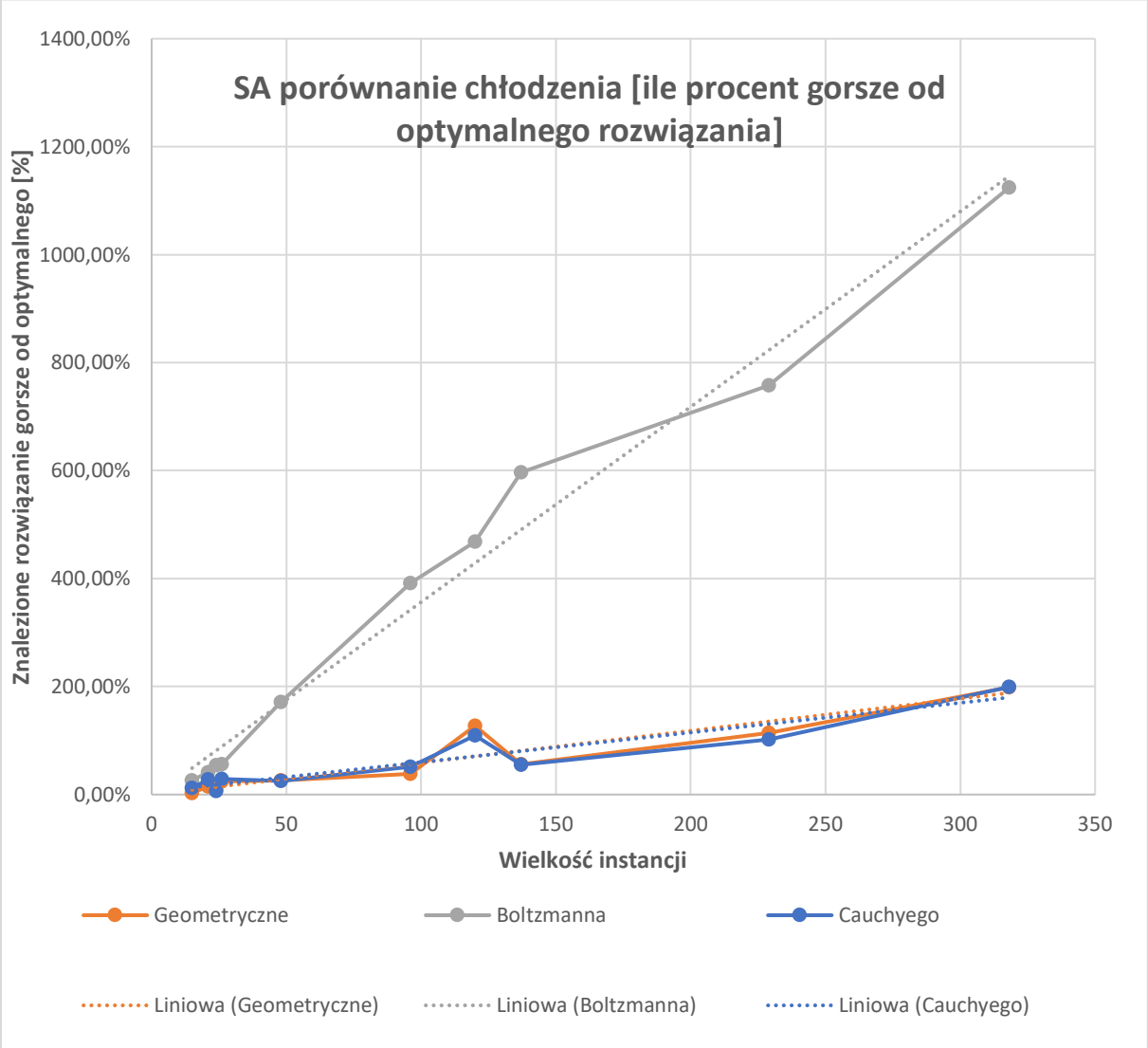
SA zbadanie wpływu sposobu generowania wartości losowych na wynik [ile procent gorsze od optymalnego rozwiązania]

Wielkość instancji	Zwykłe generowanie funkcją "rand()"	Generowanie funkcją "rand()", po wcześniejszym dorzuceniu ziarna generatora na podstawie aktualnej ilości cykli zegara od początku działania programu
6	0,00%	0,00%
10	0,00%	0,00%
12	0,00%	0,00%
13	0,00%	0,00%
14	0,00%	0,00%
15	0,00%	0,00%
21	3,92%	3,47%
24	3,85%	2,99%
26	3,42%	2,56%
48	8,68%	4,60%
96	41,54%	31,05%
120	53,54%	58,28%
137	67,13%	60,79%
229	87,35%	75,44%
318	131,69%	126,25%
431	140,62%	132,70%

Tabela 8 Dane zebrane podczas testowania jakości otrzymanego wyniku w pod wpływem zmieniającego się algorytmu generowania sąsiedztwa dla| alphaCoolingRate=0,9999 | Chłodzenia geometrycznego | Zadowalającej jakości wyniku = 5% | maxTime=300s|Długość ery = 3| Przeglądu sąsiedztwa: Dwu-zamiany

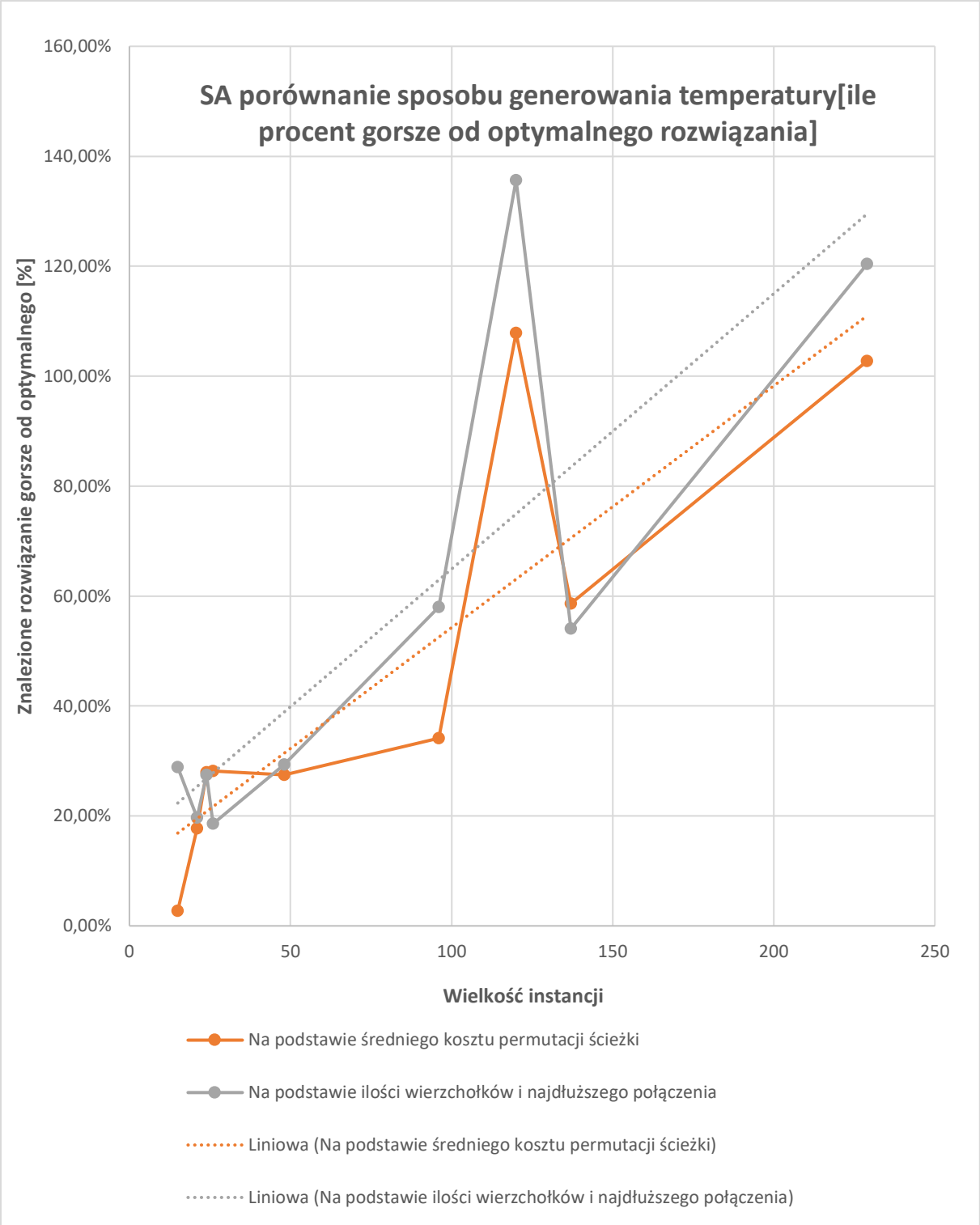
7. Analiza wyników i wnioski

Przerywaną linią zostały zaznaczone linie trendów, które pomagają lepiej zobrazować dalszy kierunek wykresu dla konkretnego parametru.



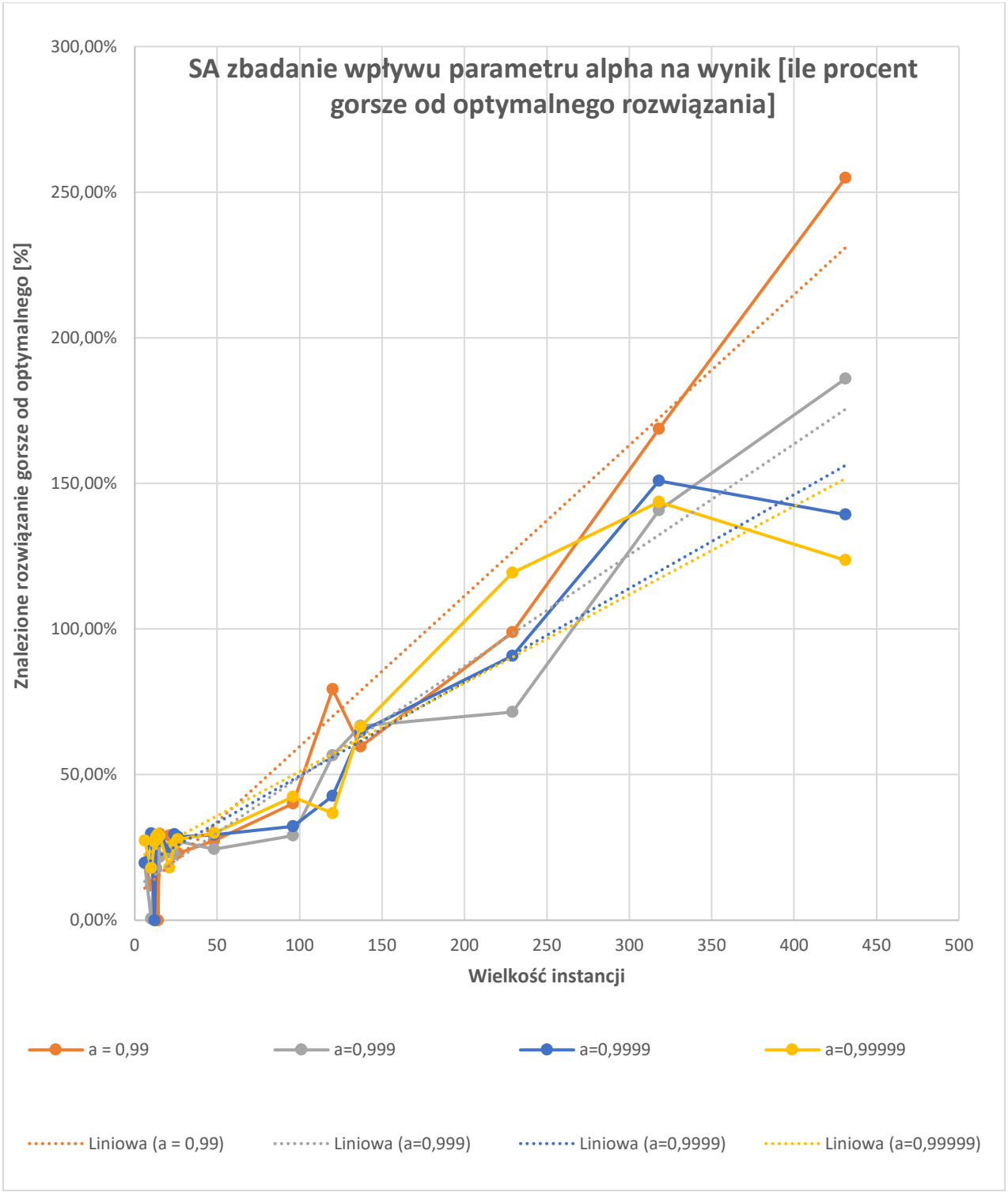
Wykres 1 – wykres dla danych z tabeli 1

Wyniki przedstawione na wykresie 1 pokazują, że nie istnieje duża różnica pomiędzy schematem chłodzenia geometryczny, a Cauchy'ego. Co ciekawe dla instancji 120 mają taki sam chwilowy wzrost błędu.



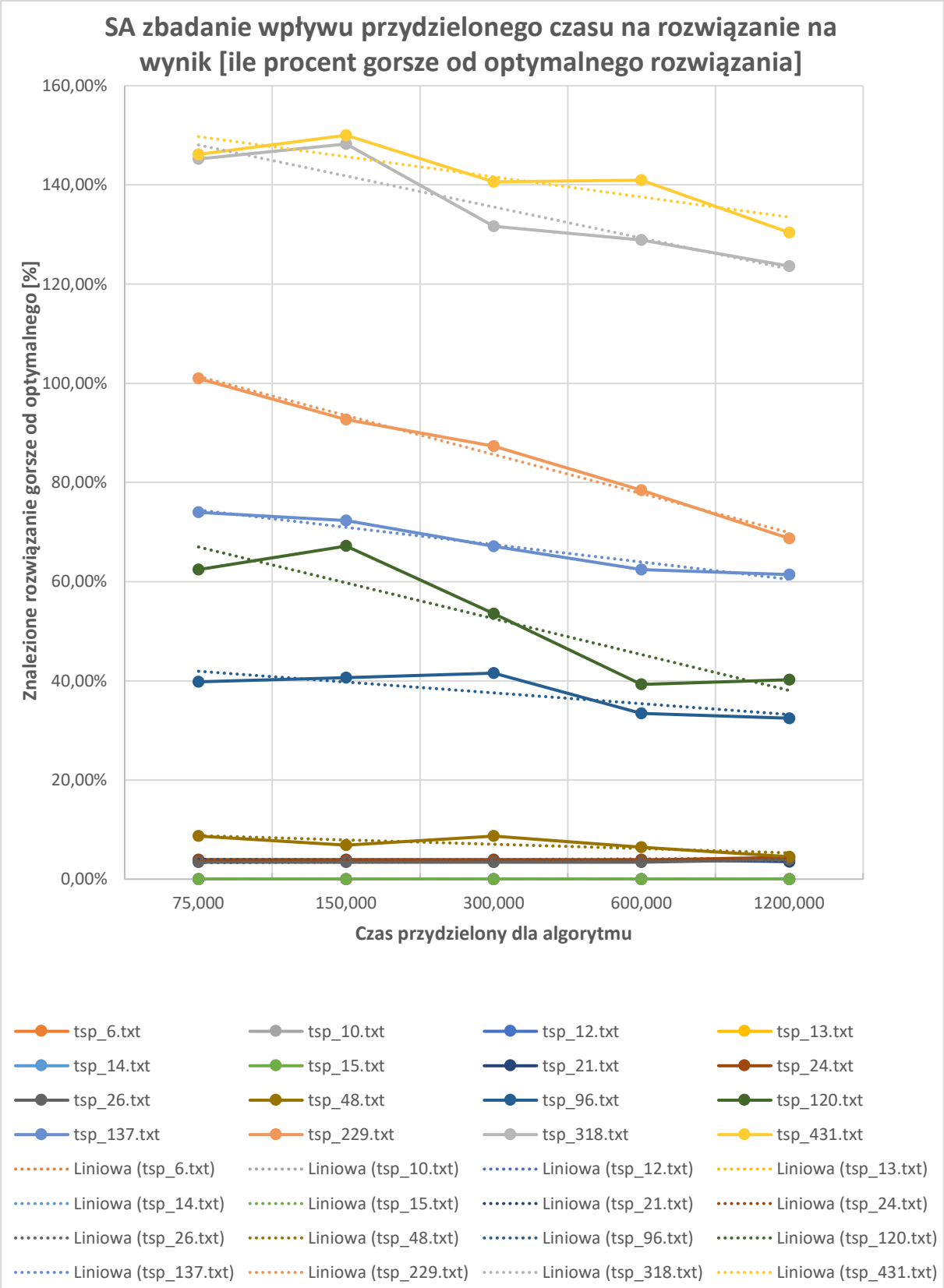
Wykres 2 – wykres dla danych z tabeli 2

Wyniki porównania sposobu generowania temperatury przedstawione na wykresie 2 wskazują jako lepszy ten opierający się na średnim koszcie permutacji. Ma to uzasadnienie w tym, że grafy prawie nigdy nie są idealnie wyważone, więc branie pojedynczej krawędzi nie daje nam praktycznie żadnych informacji o reszcie grafu/problemu.



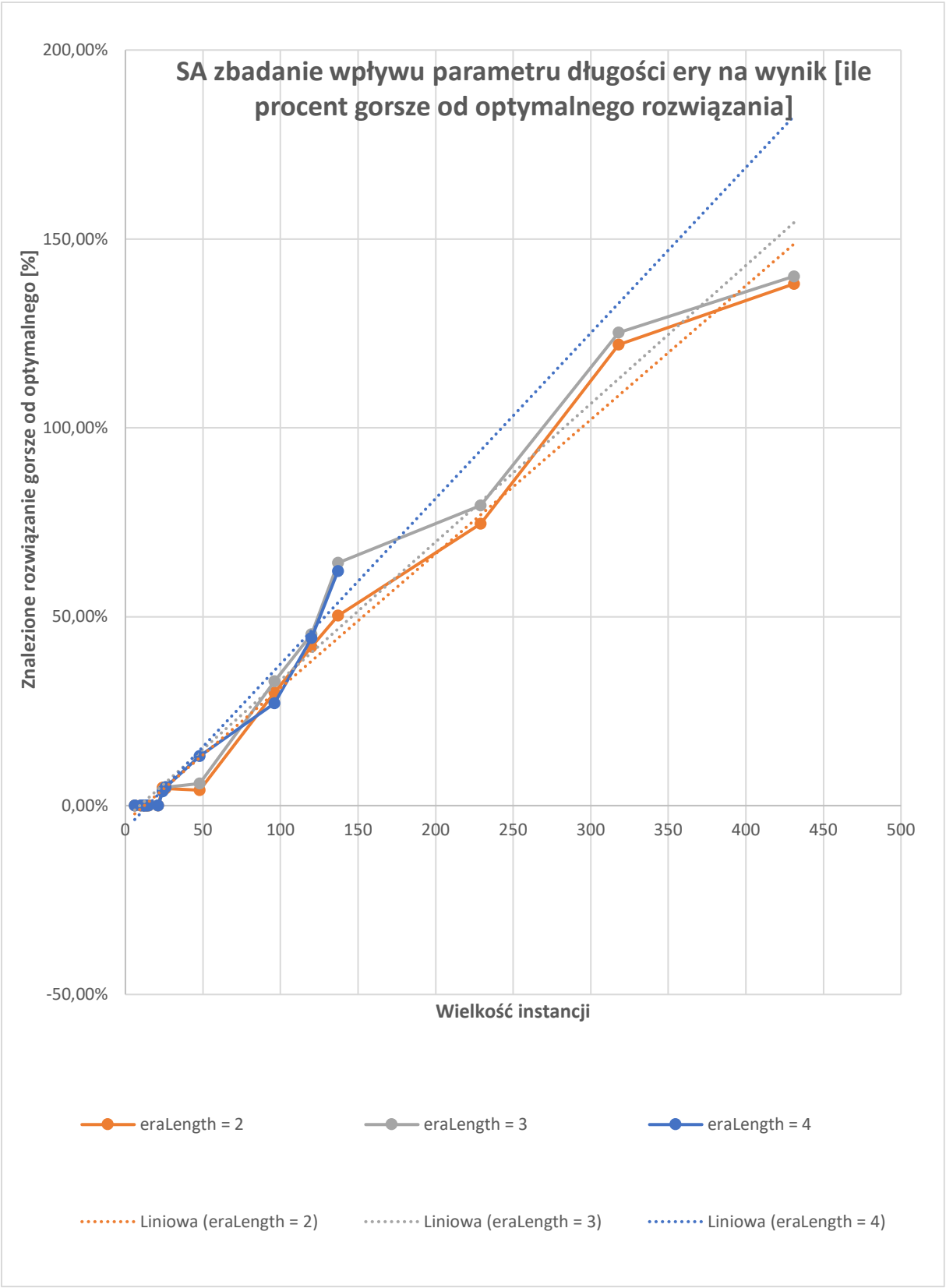
Wykres 3 – wykres dla danych z tabeli 3

Z wykresu 3 wynika, że im alpha bliższa do 1 tym algorytm dokładniejszy i generujący lepsze jakościowo rozwiązania. Jednak jest granica, gdyż pomiędzy 0,9999, a 0,99999 różnica jest już naprawdę niewielka.



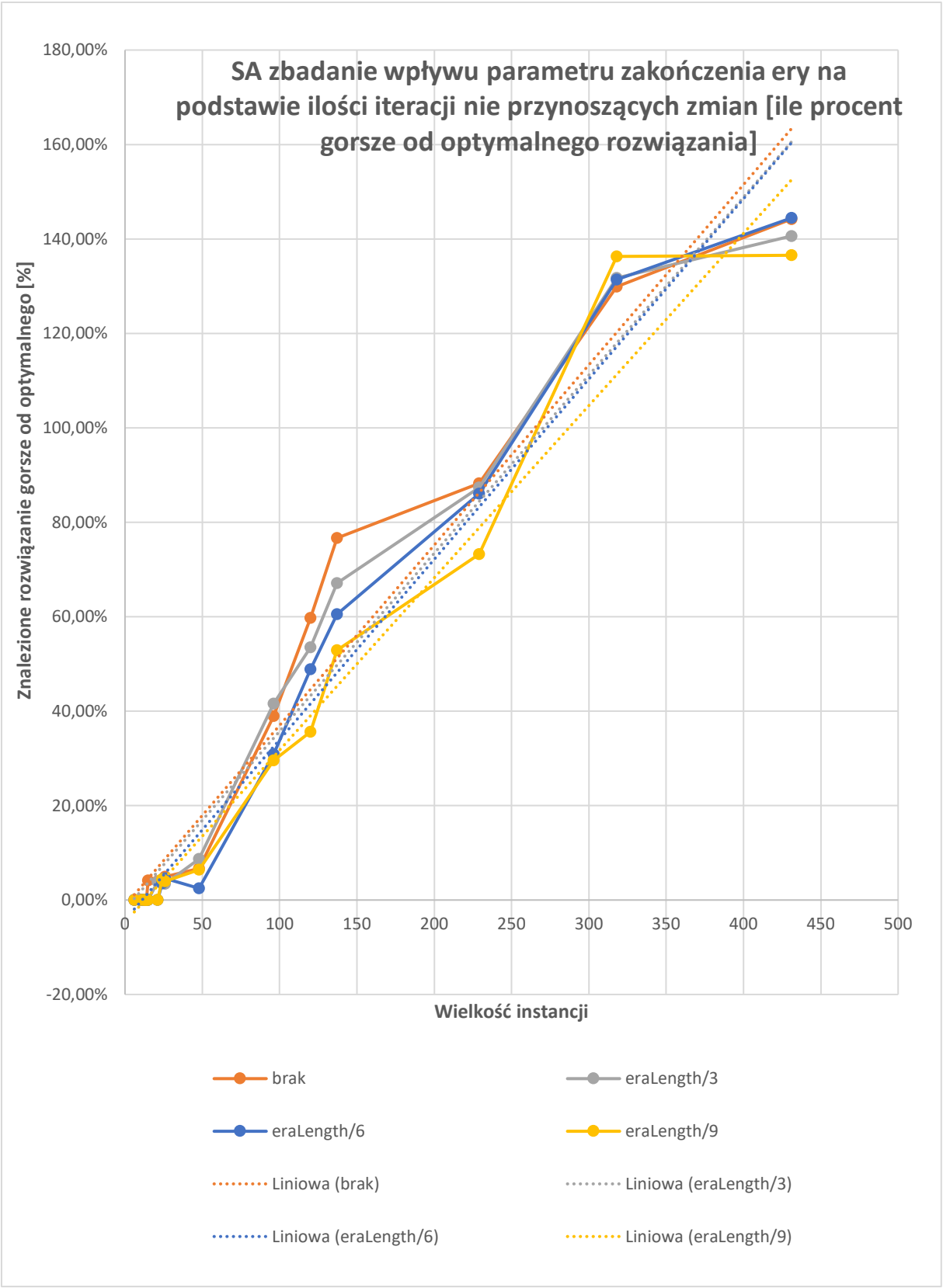
Wykres 4 – wykres dla danych z tabeli 4

Ten wykres pokazuje różnice w jakości wyniku w zależności od tego ile algorytm będzie pracować. We wszystkich instancjach jest widoczny spadek błędu wraz ze wzrostem czasu wykonywania się algorytmu, co jest bardzo prosto wytłumaczalne, gdyż jest możliwe do wykonania więcej wielostartów.



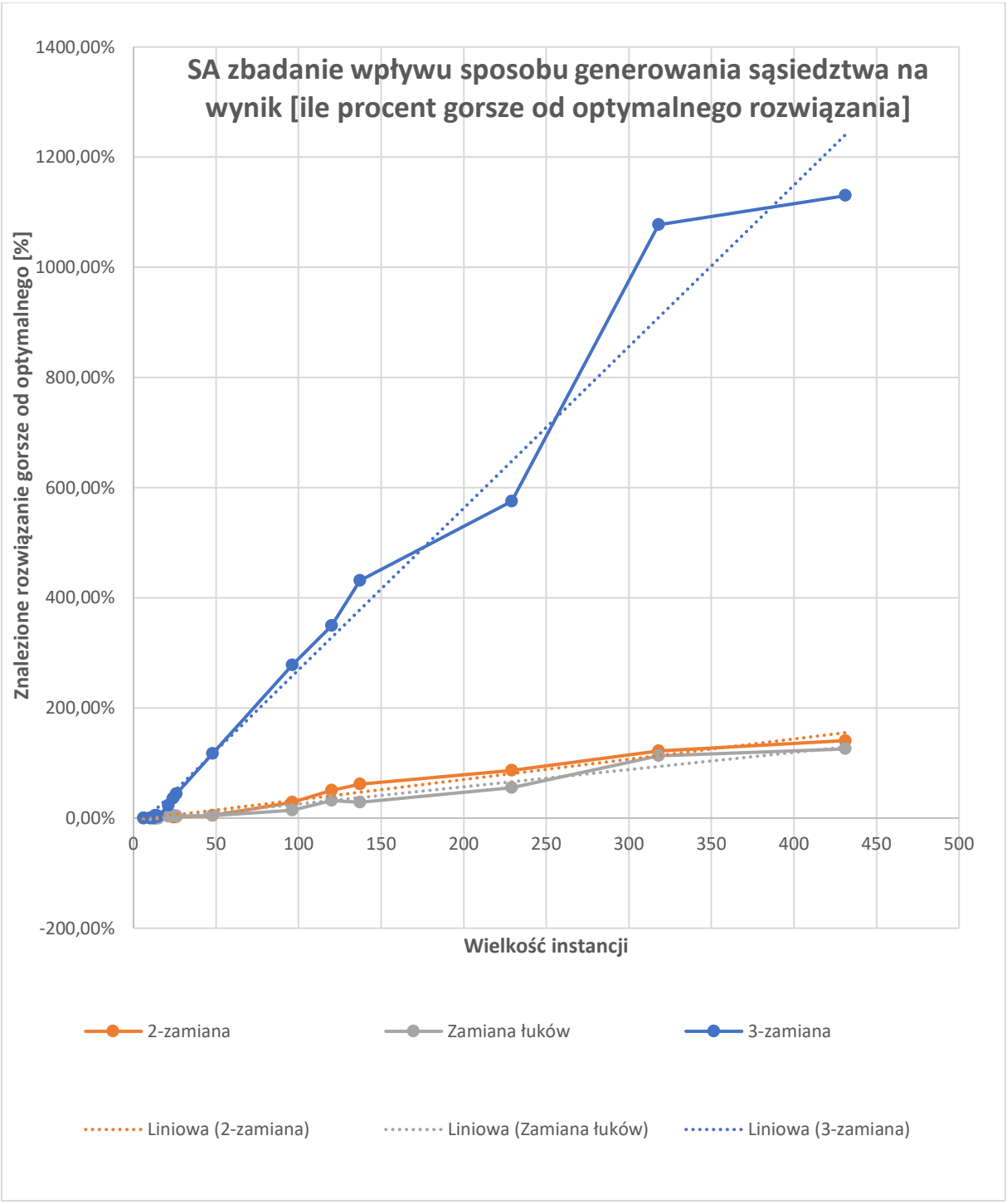
Wykres 5 – wykres dla danych z tabeli 5

Na wykresie 5 możemy zauważyć ciekawą zależność, że nie zawsze więcej znaczy lepiej. Algorytm najlepiej sobie radzi dla krótszej ery. Sugeruje to, że lepsze efekty daje krótszy czas przetwarzania jednego przypadku i skupienie się na większej ilości wielostartów, niż na odwrót.



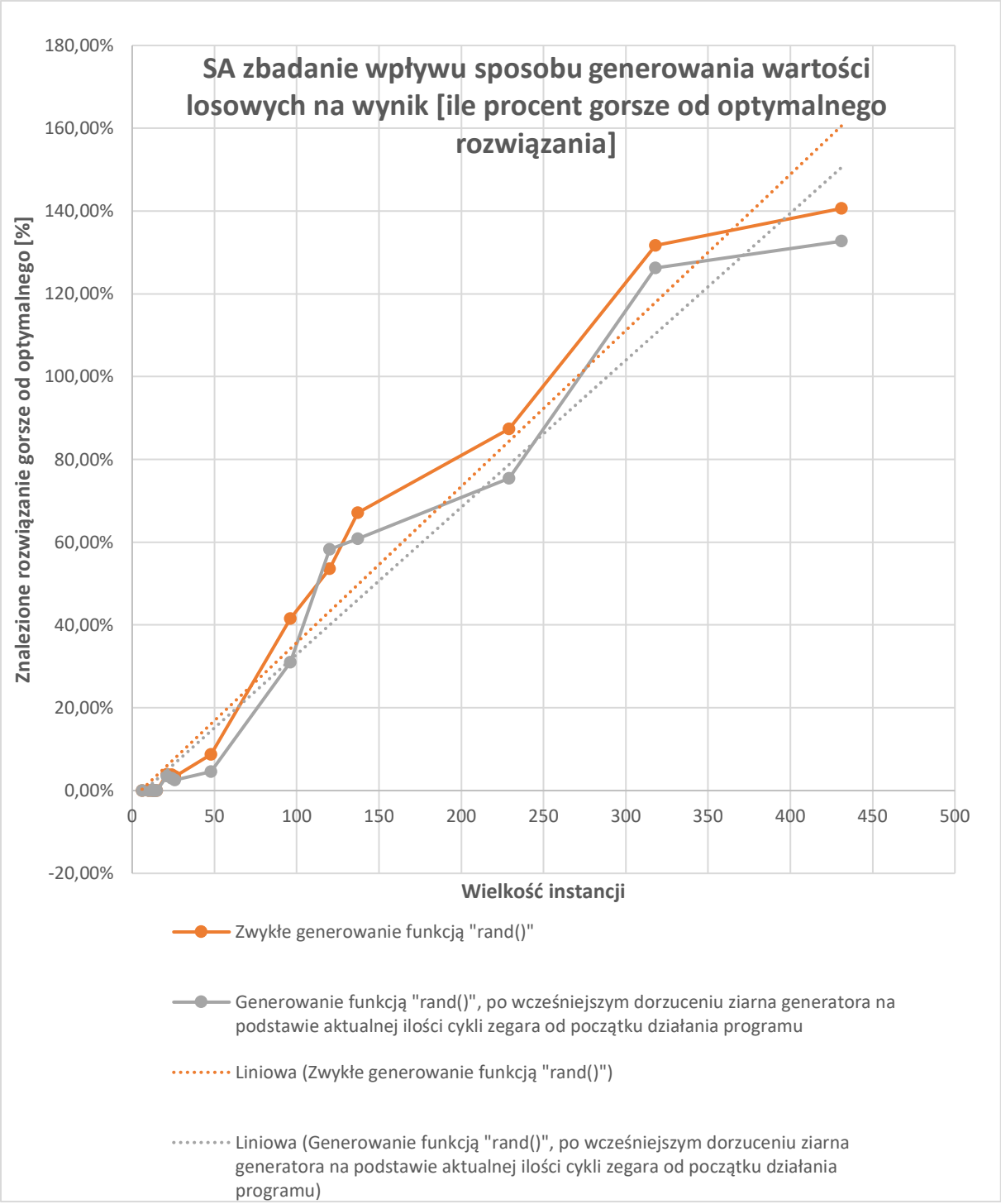
Wykres 6 – wykres dla danych z tabeli 6

Z wykresu 6 można odczytać, że im mniej szans damy algorytmowi na zrobienie jakiegoś ruchu i szybciej będziemy go przerywać, a następnie kierować się do nowej ery tym lepszy jakościowo wynik otrzymamy.



Wykres 7 – wykres dla danych z tabeli 7

Wykres 7 przedstawia skuteczność algorytmów generowania sąsiedztwa. Okazuje się, że już 3-zamiana nie jest opłacalna i 2-zamiana jest maksymalną n-zamianą, jaka ma sens. Jako najefektywniejsza metoda wychodzi tutaj zamiana łuków, czyli odwracanie części ścieżki o 180 stopni.



Wykres 8 – wykres dla danych z tabeli 8

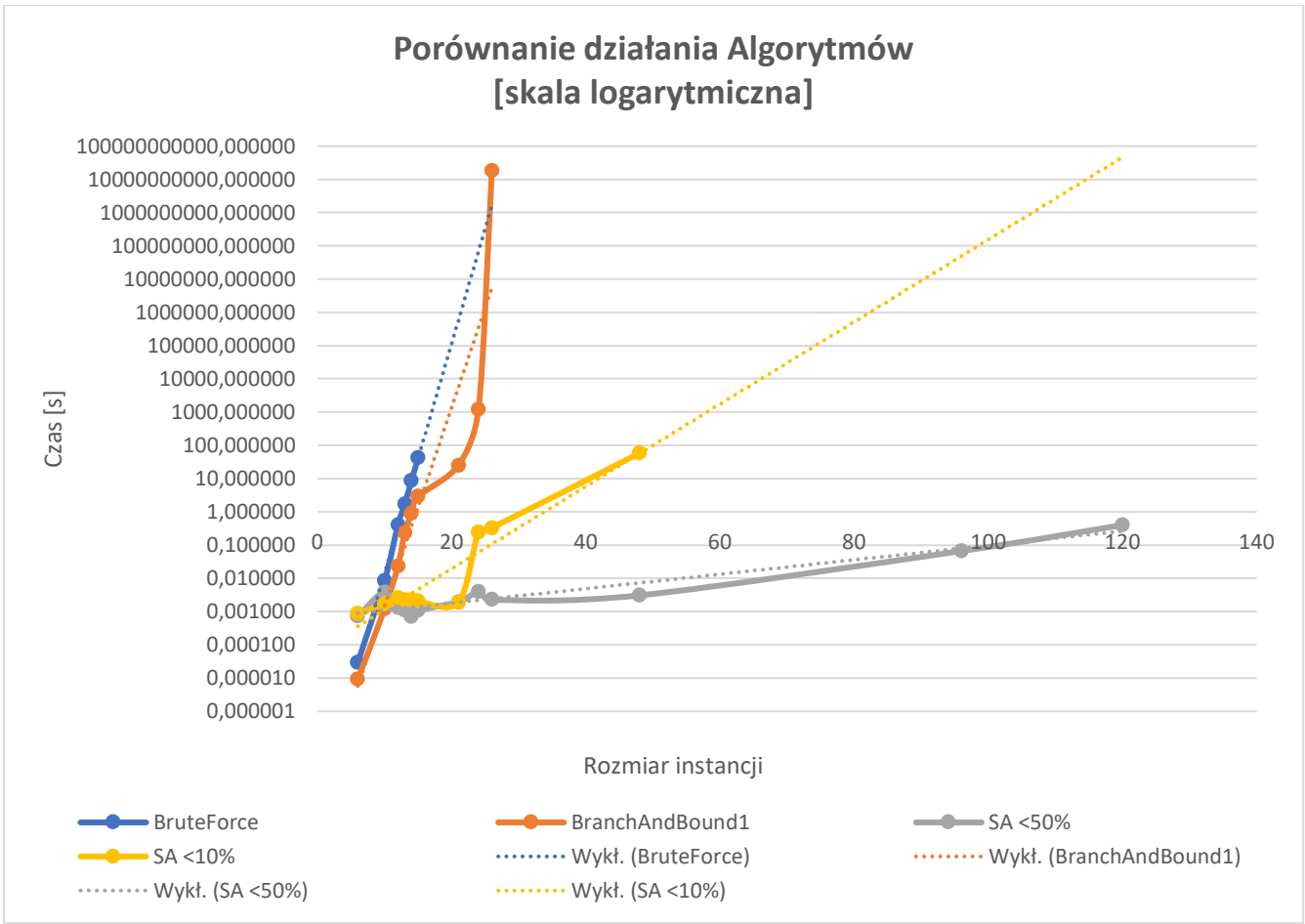
Wykres 8 bardzo dobrze uwidacznia problem jakim są liczby pseudolosowe. Żeby algorytm SA mógł działać najbardziej wydajnie powinien mieć zapewnione „idealnie losowe” liczby, co w przypadku komputera jest oczywiście niemożliwe. Na poparcie tej tezy są właśnie wyniki widoczne na wykresie 8, gdzie dodanie kolejnego elementu zwiększającego stopień losowości (jakim było podanie ziarna do funkcji „srand()” podniosło jakość wyniku.

8. Dodatek - porównanie z innymi algorytmami

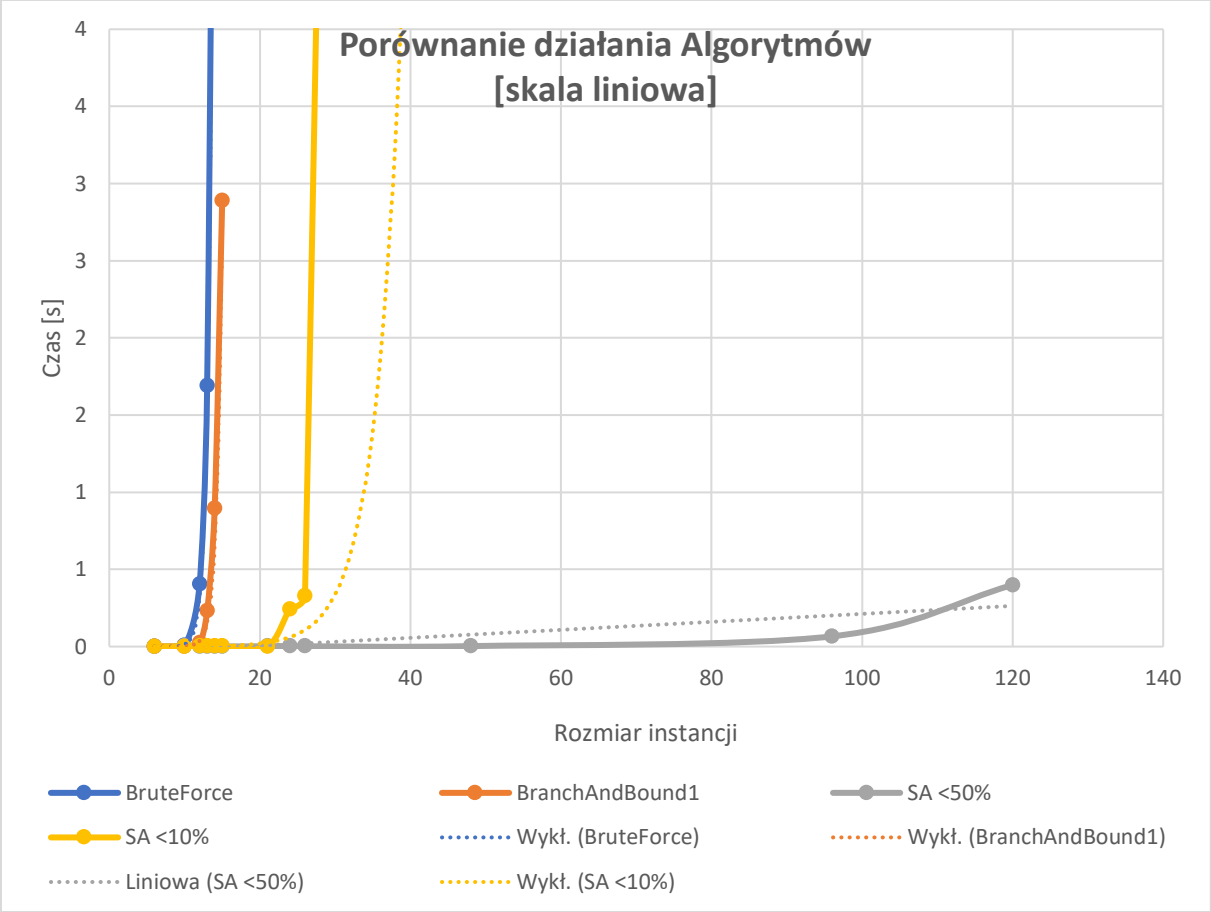
Aby porównać z innymi algorytmami ustawiłem sposób pomiaru na taki, który ma zwrócić wynik o nie gorszej jakości niż x%. W tym przypadku x = 10% oraz x = 50%. Wyniki prezentują się następująco:

PORÓWNANIE[s]				
Instancje	**BruteForce**	**BranchAndBound1**	**SA <50%**	**SA <10%**
6	0,000029151	0,000009162	0,000735	0,000862
10	0,008548054	0,001203453	0,003898	0,001707
12	0,404767974	0,023553584	0,001303	0,002573
13	1,690169648	0,230382852	0,001087	0,002244
14	8,508161312	0,896042244	0,000710	0,002263
15	41,661500140	2,889099020	0,001077	0,002068
21		24,985477000	0,001957	0,001887
24		1210,319452	0,003911	0,243829
26		18446744000	0,002271	0,32875
48			0,003071	58,25422
96			0,0662	
120			0,399185	

Tabela 9 Porównanie dotychczas przeze mnie zbadanych algorytmów



Wykres 9 - dane z tabeli 9 przedstawione na wykresie w skali logarytmicznej



Wykres 10 - dane z tabeli 9 przedstawione na wykresie w skali liniowej

Na obu wykresach widać olbrzymią przewagę Symulowanego Wyżarzania nad algorytmami dokładnymi takimi jak BruteForce czy BranchAndBound. Jest to kosztem dokładności, ponieważ aby otrzymać dużą prędkość działania musimy zrezygnować z dokładności. Widać to bardzo dobrze w różnicy pomiędzy dwoma wykresami dla SA, tylko z innym parametrem dokładności.