Projektowanie Efektywnych Algorytmów
Projekt
19/12/2023

263974 Szymon Kluska

(4) Symulowane Wyżarzanie

Spis treści	strona

1. Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności algorytmu opartego o metodę symulowanego wyżarzania rozwiązującego problem komiwojażera.
Problem komiwojażera (eng. travelling salesman problem, TSP) to zagadnienie polegające na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Cykl Hamiltona to cykl w grafie, w którym każdy wierzchołek grafu jest odwiedzany dokładnie raz oprócz wierzchołka startowego. Rozwiązanie polega na znalezieniu w grafie ścieżki o najmniejszym koszcie.

Hipotezy
Metoda symulowanego wyżarzania jest metodą heurystyczną co oznacza, że dąży do znalezienia rozwiązania o wartości funkcji celu jak najbliższej wartości optymalnej, ale nie gwarantuje znalezienia optymalnego rozwiązania. Wyniki mogą się różnić w zależności od wybranych parametrów.
W trakcie przebiegu algorytmu do pamięci nie zostają zapisane informacje o poprzednich iteracjach, więc zużycie pamięci powinno być podobne dla każdej wielkości instancji.
Czas wykonania programu będzie zależny od wielkości badanej instancji oraz sposobu doboru długości epoki.

2. Metoda

Symulowane wyżarzanie (ang. Simulated Annealing) to metaheurystyka oparta na analogii zaczerpniętej z termodynamiki. Proces wyżarzania w fizyce polega na podgrzaniu, a następnie powolnym chłodzeniu substancji w celu uzyskania silnej struktury krystalicznej. Jeśli temperatura początkowa nie jest wystarczająco wysoka lub stosuje się zbyt szybkie chłodzenie, uzyskuje się niedoskonałości. W takim przypadku chłodzone ciało stałe nie osiągnie równowagi termicznej w każdej temperaturze. Silne kryształy powstają w wyniku ostrożnego i powolnego chłodzenia. Metoda symulowanego wyżarzania symuluje ten proces.

W metodzie symulowanego wyżarzania występuje parametr sterujący T, nazywany temperaturą (tak jak w prawdziwym wyżarzaniu). Im wyższa jest wartość temperatury tym jest większe prawdopodobieństwo zaakceptowania gorszego rozwiązania.

Aby uniknąć dążenia tylko do optimum lokalnego zamiast do optimum globalnego, metoda symulowanego wyżarzania pozwala na akceptowanie gorszych rozwiązań bazując na prawdopodobieństwie $P = e^{\frac{f(x)-f(x')}{T}}$ (zasada akceptacji Metropolisa), gdzie x jest obecnym rozwiązaniem, x' sąsiadem, a f to funkcja celu (w przypadku TSP kosztu drogi).
Jeżeli $P \geq R$, gdzie R to losowa wartość z przedziału <0, 1> to gorsze rozwiązanie zostaje zaakceptowane jako obecne.
Wpływ na metodę SA mają również parametry temperatury początkowej, doboru rozwiązania początkowego, doboru rozwiązania w sąsiedztwie, schematu chłodzenia oraz kryteriów zakończenia.

Temperatura początkowa

Jednym z sposobów wyznaczania temperatury początkowej jest wygenerowanie 100 sąsiadów dla rozwiązania początkowego, a następnie wyznaczenie wartości bezwzględnej z różnic wartości funkcji celu. Następnie wybiera się parametr τ w zależności od zakładanej jakości początkowej konfiguracji

- τ = 50% jeśli zakładana jakość jest słaba (co powoduje rozpoczęcia od wyższej temperatury)
- $\tau = 20\%$ jeśli zakładana jakość jest dobra (co powoduje rozpoczęcia od niższej temperatury)

Następnie oblicza się temperaturę początkową T_0 z wzoru

$$T_0 = \frac{-\Delta}{\log(\tau)}$$

Gdzie Δ to średnia wyznaczonych wartości bezwzględnych z różnic wartości funkcji celu. [1]

Schematy chłodzenia
Geometryczny

$$T_{k+1} = \lambda T_k$$

gdzie $\lambda \leq 1$

Logarytmiczny (Boltzmanna)

$$T_{k+1} = \frac{T_k}{(1 + \lambda * T_k)}$$
$$\lambda = \frac{T_0 - T_N}{N * T_0 * T_N}$$

gdzie
N – maksymalna ilość iteracji
T_0 – temperatura początkowa
T_N – temperatura końcowa

Dobór długości epoki

W implementacji algorytmu opartego o metodzie SA zostały użyte dwa sposoby dobierania długości epok dla sąsiedztwa o wielkości n:

- $x * n$, gdzie x to parametr podany przez użytkownika
- n^2

Dobór rozwiązania początkowego

Rozwiązanie początkowe jest losowane spośród wszystkich możliwych rozwiązań.

Kryterium zakończenia

Kryterium zakończenia wykonywania SA mogą być

- Określona liczba iteracji
- Brak poprawy po określonej liczbie iteracji
- Liczba zaakceptowanych gorszych rozwiązań poniżej ustalonego poziomu
- Akceptowalny poziom błędu na bazie oszacowań LB i UB.
- Przekroczony ustalony czas wykonania dla danego poziomu temperatury
- Temperatura poniżej ustalonej temperatury minimalnej

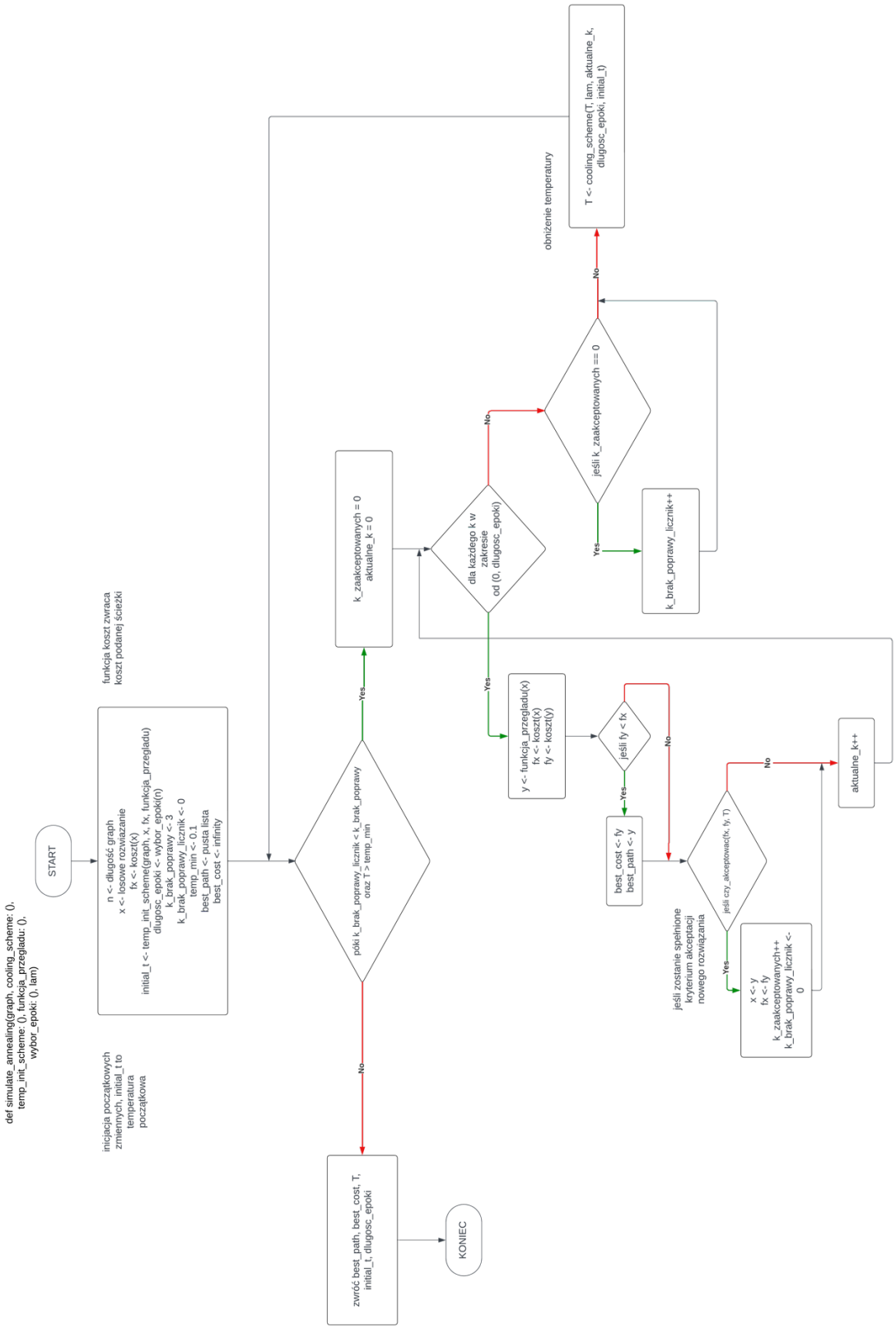
W implementacji zostały zastosowane kryteria:

- Brak poprawy po 3 iteracjach [1]
- Temperatura poniżej temperatury minimalnej równej 0.1

Dobór rozwiązania w sąsiedztwie

- 2-zamiana
  N(x) to zbiór rozwiązań powstałych przez usunięcie 2 miast i wstawienie ich w inne miejsca tworząc nową permutacje o innej kolejności miast.
- Wymiana łuków
  N(x) to zbiór rozwiązań powstałych przez usunięcie k kolejnych miast i wstawienie ich w odwrotnej kolejności

3. Algorytm



Rysunek 1 Schemat blokowy algorytmu dla metody symulowanego wyżarzania

4. Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw danych instancji:

1. tsp_6_1.txt 20 132 [0, 1, 2, 3, 4, 5, 0]
2. tsp_6_2.txt 20 80 [0, 5, 1, 2, 3, 4, 0]
3. tsp_10.txt 20 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
4. tsp_12.txt 20 264 [0, 1, 8, 4, 6, 2, 11, 9, 7, 5, 3, 10, 0]

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

Do wykonania badań wybrano następujący zestaw instancji:

1. tsp_6_1.txt 20 132 [0, 1, 2, 3, 4, 5, 0]
2. tsp_6_2.txt 20 80 [0, 5, 1, 2, 3, 4, 0]
3. tsp_10.txt 20 212 [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
4. tsp_12.txt 20 264 [0, 1, 8, 4, 6, 2, 11, 9, 7, 5, 3, 10, 0]
5. tsp_13.txt 20 269 [0, 10, 3, 5, 7, 9, 11, 2, 6, 4, 8, 1, 12, 0]
6. tsp_14.txt 20 282 [0, 10, 3, 5, 7, 9, 13, 11, 2, 6, 4, 8, 1, 12, 0]
7. tsp_15.txt 20 291 [0, 10, 3, 5, 7, 9, 13, 11, 2, 6, 4, 8, 14, 1, 12, 0]
8. tsp_17.txt 20 39 [0, 2, 13, 1, 9, 10, 12, 5, 6, 14, 15, 3, 4, 8, 16, 7, 11, 0]

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

9. gr96.txt 20 55209
10. kroa100.txt 20 21282
11. krob150.txt 20 22141
12. pr152.txt 20 73682
13. krob200.txt 20 29437
14. rgb323.txt 20 1326

http://softlib.rice.edu/pub/tsplib/tsp/

Dla powyższych danych z tsplib na stronie podane są tylko optymalne koszty (bez ścieżek).
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf

5. Procedura badawcza

Należało zbadać wpływ sposobu dobierania temperatury początkowej T_0, sposobu wyboru rozwiązania początkowego x_0, sposobu dobierania schematu chłodzenia, sposobu dobierania długości epoki i sposobu wyboru rozwiązania w sąsiedztwie rozwiązania bieżącego na jakość uzyskiwanych rozwiązań (wielkość błędu), czas uzyskiwania rozwiązań oraz zużycie pamięci. Procedura badawcza polegała na uruchomieniu programu sterowanego plikiem konfiguracyjnym test_atsp.ini. Dla każdego pliku testowego dla każdej konfiguracji parametrów metoda symulowanego wyżarzania została wykonana 20 razy, tak aby czas dla pojedynczej konfiguracji nie przekraczał ustalonego czasu (około godziny).
Program został napisany w języku Python 3.10, a generowanie wyników procedury badawczej zostało przeprowadzone na MacBooku Pro o procesorze Apple M1, 16GB RAM i systemie macOS Sonoma 14.1.
Do pomiaru czasu została użyta biblioteka time [2], a do pomiaru zużycia pamięci biblioteka memory_profiler [3].

```
from time import perf_counter
graph = file_to_graph(file_name)

start = perf_counter()
memory_profiler.profile()
path, dist, temp_koncowa, temp_poczatkowa, ile_epok = method(graph, chlodzenie[parametry[1]],
                                                             wybor_t0[parametry[0]], sposob[parametry[3]],
                                                             epoki[parametry[2]])
mem_usage = memory_profiler.memory_usage()
stop = perf_counter()
diff = stop - start
return path, dist, diff, mem_usage, ile_epok, temp_poczatkowa, temp_koncowa
```

Rysunek 2 Fragment kodu funkcji perform_method_sa odpowiadający za pomiar czasu i pamięci w module pea_utils.py

Do pliku wyjściowego test_atsp_out.csv najpierw zapisywane były: nazwa pliku zestawu instancji, ilość powtórzeń i wybrane parametry. Następnie program zapisywał otrzymywane wyniki: ścieżkę, koszt wykonania ścieżki, czas wykonania funkcji algorytmu [w sekundach], zużycie pamięci [w MiB], długość epoki, temperaturę początkową i temperaturę końcową. Plik wyjściowy był zapisywany w formacie csv.
Poniżej przedstawiono fragment zawartości przykładowego pliku wyjściowego.
dane/tsp_6_1.txt,20,132,"'temp_wzor', 'geometryczny', 'mnoznik', 'luk'"
"0, 1, 2, 3, 4, 5, 0", 132,0.10922616699826904, 45.6875, 60, 25.564556124552432, 6.49816716293987
"0, 1, 2, 3, 4, 5, 0", 132, 0.11034054099582136, 45.703125,60, 6.896082295449245, 4.072067634639825
"0, 1, 2, 3, 4, 5, 0", 132, 0.10366329099633731, 45.703125,60, 4.4579276763468965, 2.9248463484511995
…
test_wzor_los_geo_mnoz_luk.csv

Wyniki zostały opracowane przy użyciu biblioteki pandas [7] i programu Microsoft Excel.
Przy badaniu wpływu doboru poszczególnych parametrów były ustawiane standardowe parametry (schemat geometryczny, długość epoki n*10, wymiana łuków) z zmieniony badanymi parametrami np. dla badania doboru rozwiązania badanymi konfiguracjami były „schemat geometryczny, długość epoki n*10, wymiana łuków” i „schemat geometryczny, długość epoki n*10, 2-zamiana”.
Wielkość błędu względem optimum jest wyliczana ze wzoru

$$błąd = \frac{otrzywany\ wynik - optimum}{optimum} * 100\%$$

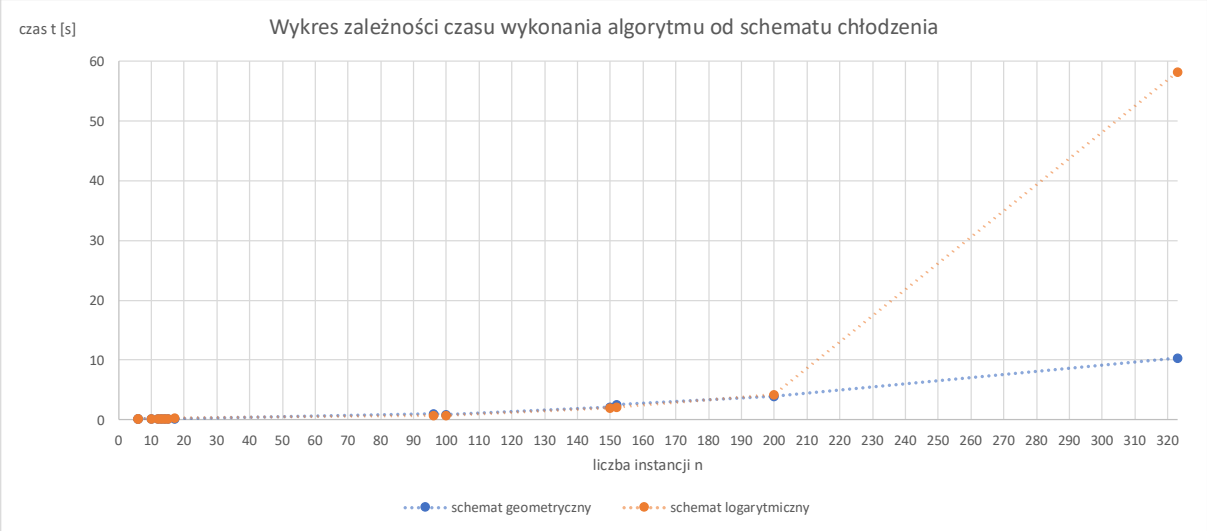
6. Wyniki

Wyniki przedstawione zostały w postaci wykresu zależności czasu uzyskania rozwiązania problemu od wielkości instancji i tabeli z dokładnymi wartościami.

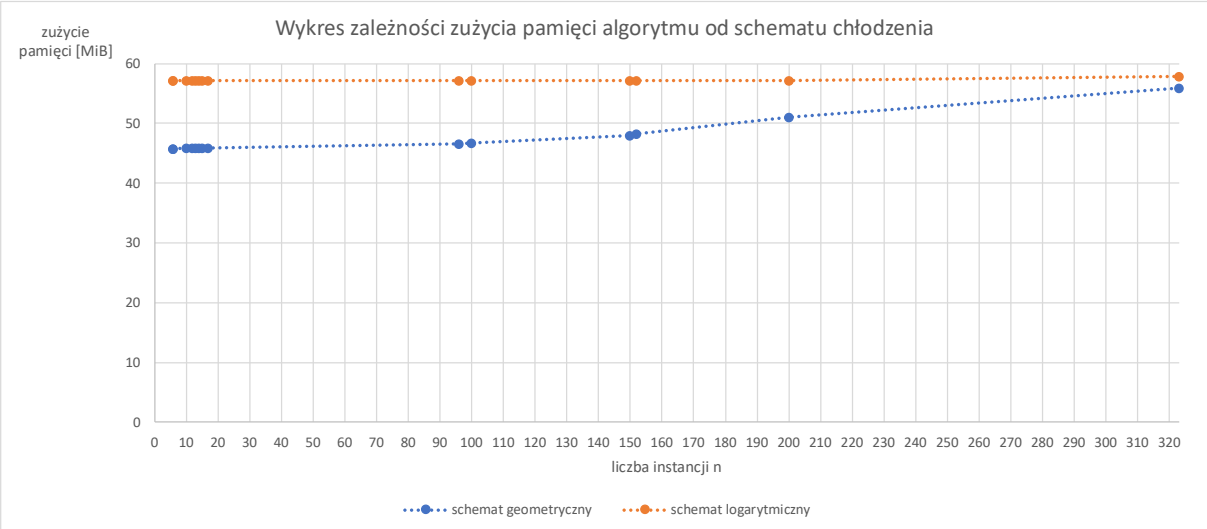
Wyniki wyboru różnych schematów chłodzenia

Tabela 1 Wyniki badanego czasu, zużycia pamięci i średniej wysokości błędu dla schematu chłodzenia geometrycznego i logarytmicznego

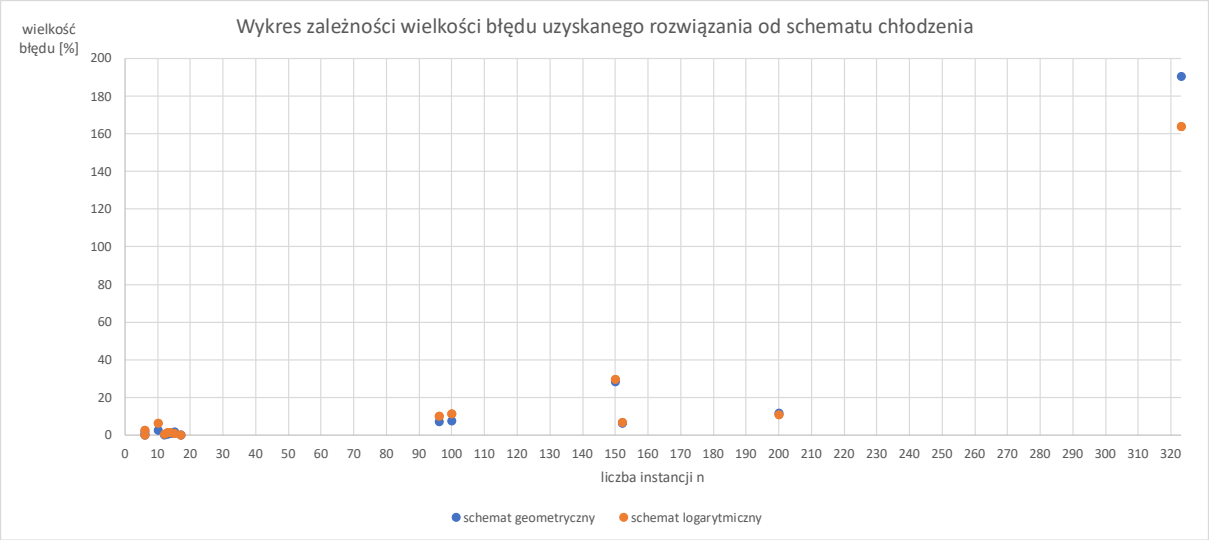
	geometryczny			logarytmiczny		
n	średni czas [s]	średnie zużycie pamięci [MiB]	średnia wysokość błędu [%]	średni czas [s]	średnie zużycie pamięci [MiB]	średnia wysokość błędu [%]
6	0,1148	45,73	1,29	0,1188	57,11	2,58
6	0,1273	45,76	0,00	0,1220	57,11	0,00
10	0,1530	45,82	2,62	0,1792	57,11	6,37
12	0,1541	45,84	0,00	0,1989	57,11	0,42
13	0,1567	45,86	0,41	0,2078	57,11	1,23
14	0,1597	45,88	0,74	0,2235	57,11	1,38
15	0,1573	45,88	1,44	0,2079	57,11	0,82
17	0,1619	45,88	0,00	0,2985	57,11	0,00
96	0,9803	46,55	6,98	0,7323	57,11	10,11
100	0,8415	46,71	7,23	0,6822	57,11	11,15
150	2,1272	48,03	28,40	1,9629	57,11	29,62
152	2,5040	48,20	6,23	2,0852	57,11	6,71
200	3,8666	51,01	11,67	4,1788	57,14	10,85
323	10,3837	55,95	190,55	58,2861	57,79	163,72



Wykres 1 Wykres czasu wykonania algorytmu dla różnych schematów chłodzenia



Wykres 2 Wykres złożoności czasowej algorytmu dla różnych schematów chłodzenia

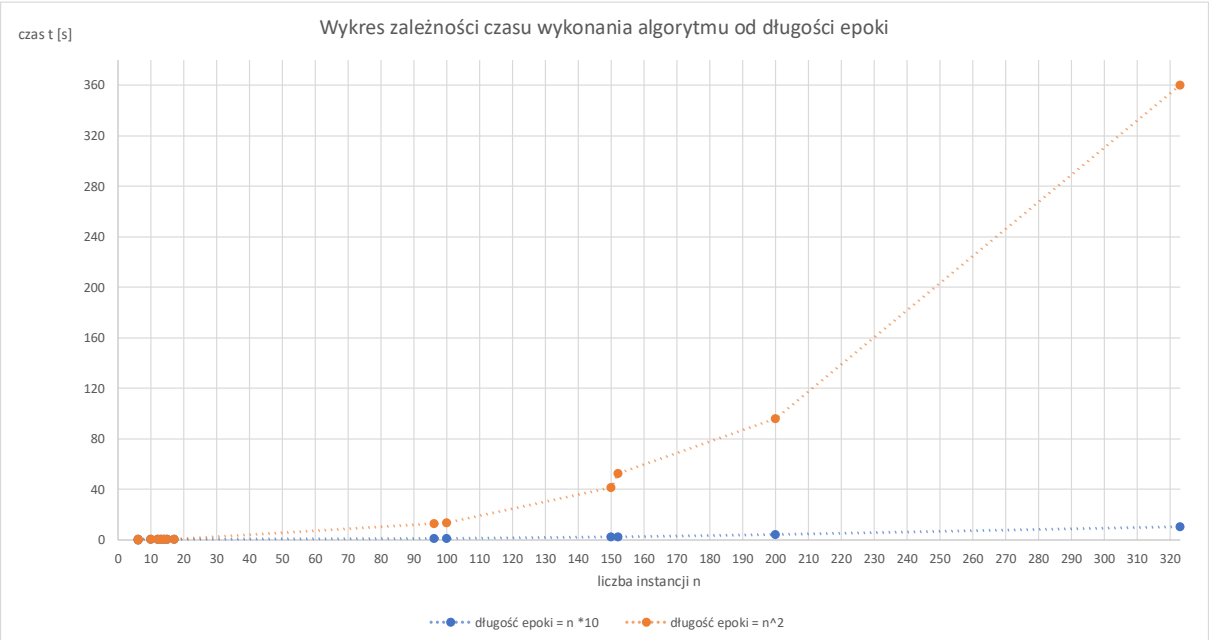


Wykres 3 Wykres średniej wielkości błędu dla różnych schematów chłodzenia

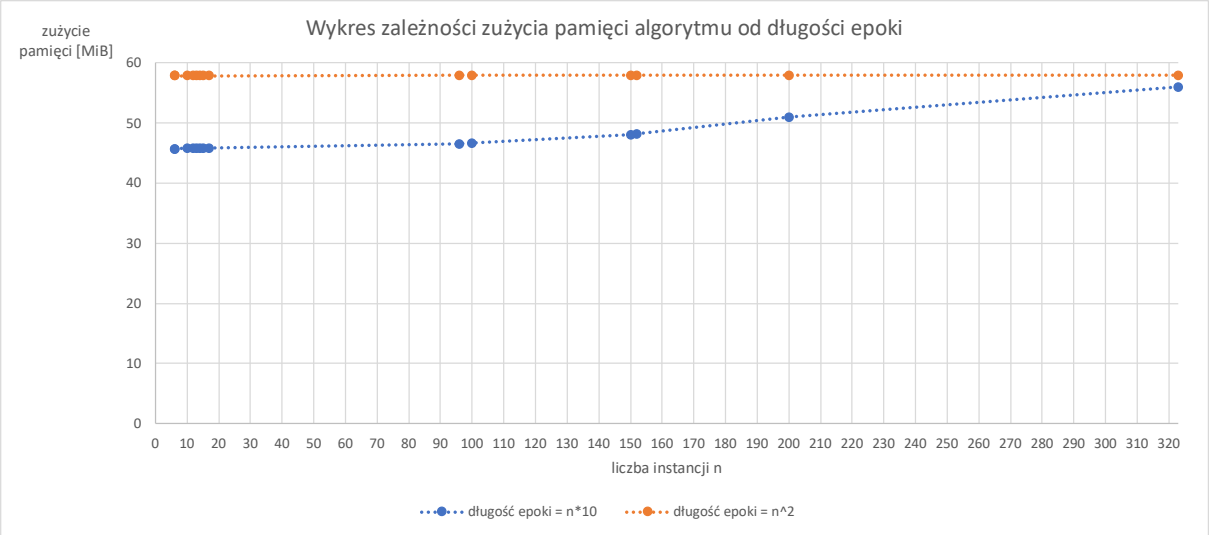
Wyniki wyboru różnych sposobów dobierania długości epoki

Tabela 2 Wyniki badanego czasu, zużycia pamięci i średniej wysokości błędu dla różnego dobierania długości epoki

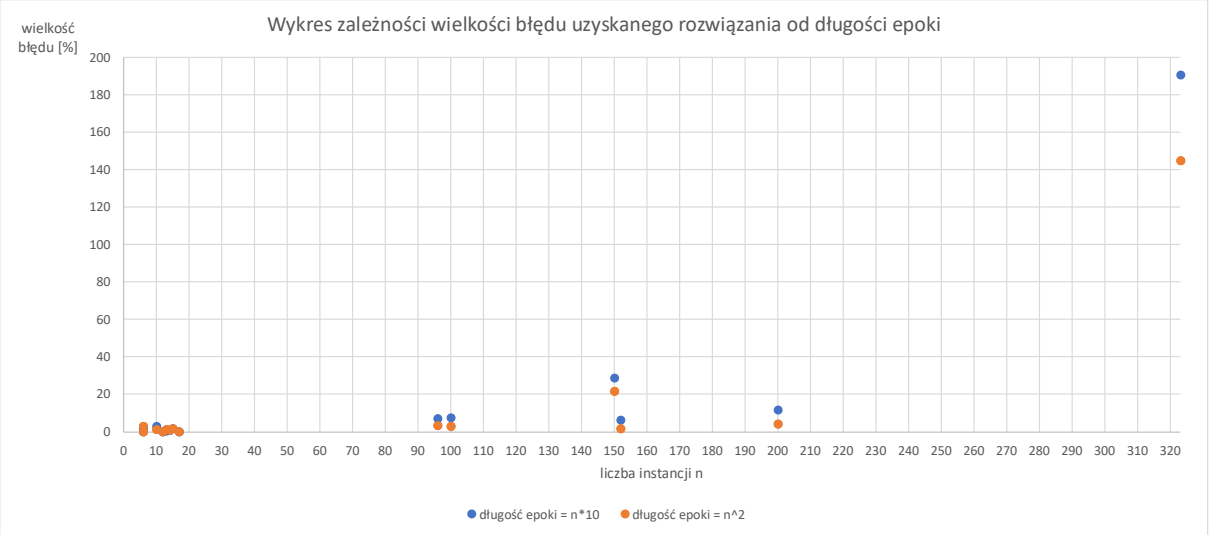
	n*10			n^2		
n	średni czas [s]	średnie zużycie pamięci [MiB]	średnia wysokość błędu [%]	średni czas [s]	średnie zużycie pamięci [MiB]	średnia wysokość błędu [%]
6	0,1148	45,73	1,29	0,1179	57,95	0,00
6	0,1273	45,76	0,00	0,1260	57,95	2,56
10	0,1530	45,82	2,62	0,1598	57,95	0,97
12	0,1541	45,84	0,00	0,1665	57,95	0,00
13	0,1567	45,86	0,41	0,1671	57,95	1,23
14	0,1597	45,88	0,74	0,1717	57,95	0,64
15	0,1573	45,88	1,44	0,1802	57,95	1,44
17	0,1619	45,88	0,00	0,1989	57,95	0,00
96	0,9803	46,55	6,98	12,9935	57,97	3,26
100	0,8415	46,71	7,23	13,7154	57,97	2,56
150	2,1272	48,03	28,40	41,5156	57,97	21,39
152	2,5040	48,20	6,23	52,7804	57,97	1,44
200	3,8666	51,01	11,67	95,9668	57,97	3,96
323	10,3837	55,95	190,55	360,0812	57,97	144,58



Wykres 4 Wykres czasu wykonania algorytmu dla różnych sposobów dobierania długości epoki



Wykres 5 Wykres złożoności czasowej algorytmu dla różnych sposobów dobierania długości epoki

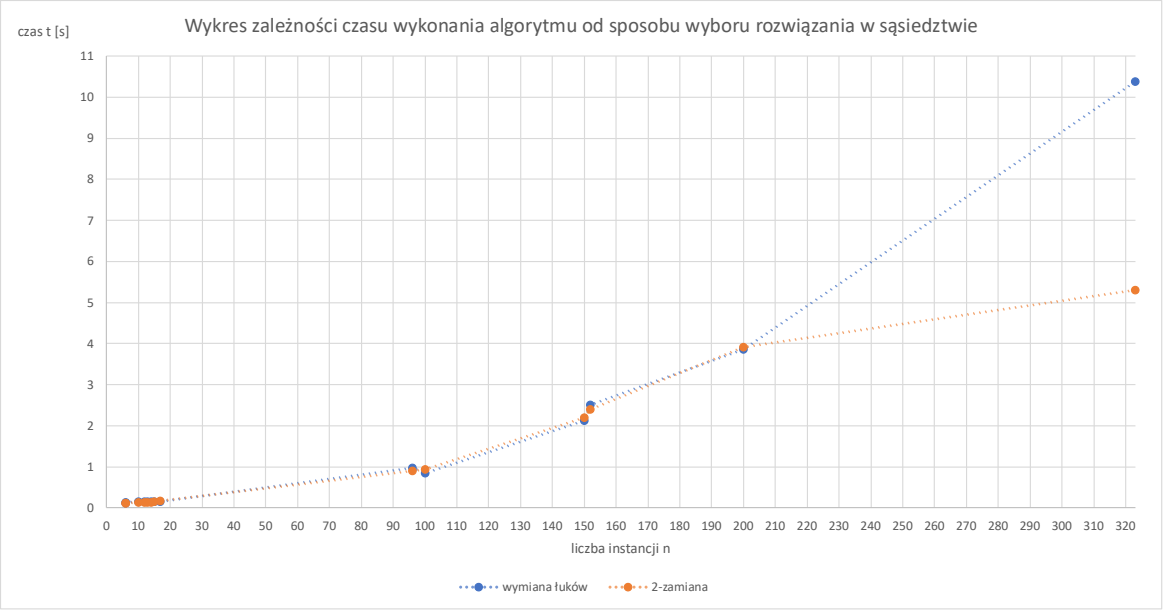


Wykres 6 Wykres średniej wielkości błędu dla różnych sposobów dobierania długości epoki

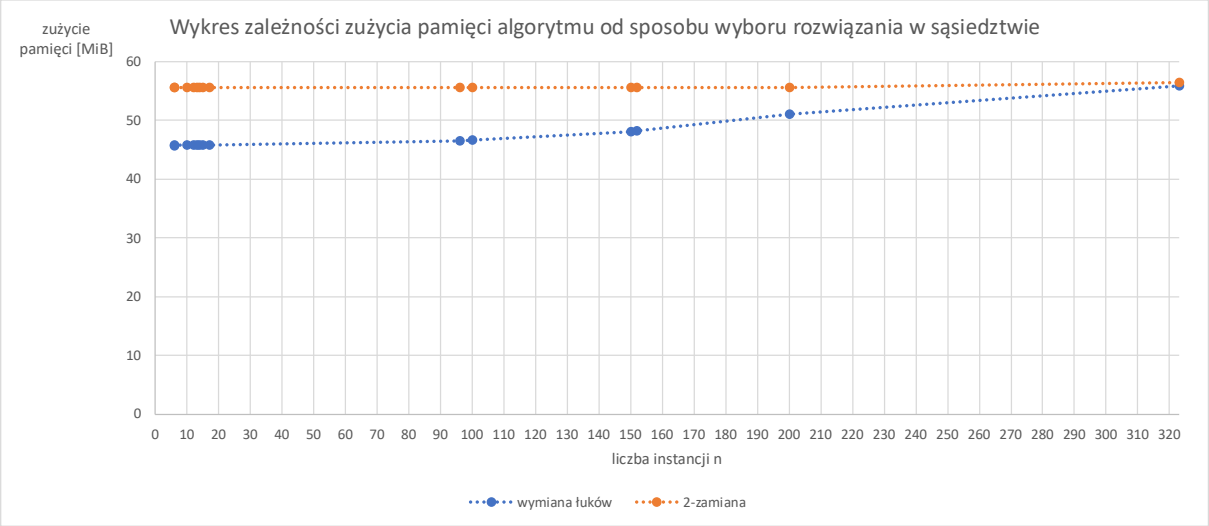
Wyniki wyboru różnych sposobów wyboru rozwiązania w sąsiedztwie

Tabela 3 Wyniki badanego czasu, zużycia pamięci i średniej wysokości błędu dla różnych sposobów doboru sąsiedztwa

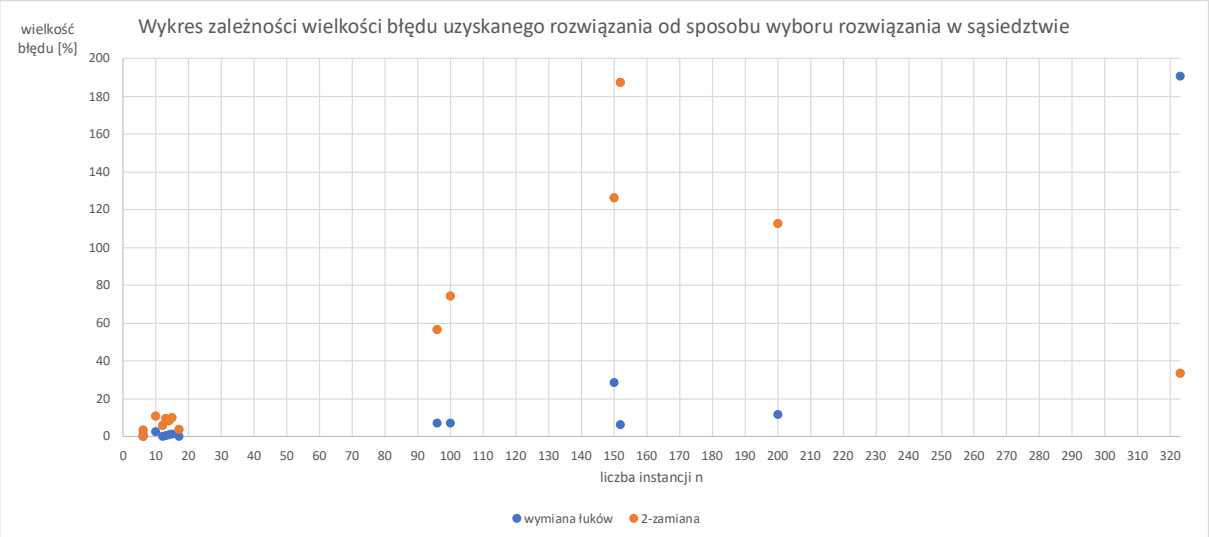
	wymiana łuków			2-zamiana		
n	średni czas [s]	średnie zużycie pamięci [MiB]	średnia wysokość błędu [%]	średni czas [s]	średnie zużycie pamięci [MiB]	średnia wysokość błędu [%]
6	0,1148	45,73	1,29	0,1187	55,57	0,00
6	0,1273	45,76	0,00	0,1253	55,62	3,25
10	0,1530	45,82	2,62	0,1412	55,63	10,57
12	0,1541	45,84	0,00	0,1345	55,63	5,98
13	0,1567	45,86	0,41	0,1387	55,63	9,44
14	0,1597	45,88	0,74	0,1390	55,63	8,14
15	0,1573	45,88	1,44	0,1475	55,63	9,85
17	0,1619	45,88	0,00	0,1647	55,63	3,72
96	0,9803	46,55	6,98	0,9066	55,64	56,43
100	0,8415	46,71	7,23	0,9407	55,64	74,33
150	2,1272	48,03	28,40	2,2026	55,66	126,14
152	2,5040	48,20	6,23	2,3920	55,66	187,37
200	3,8666	51,01	11,67	3,9164	55,66	112,76
323	10,3837	55,95	190,55	5,3077	56,49	33,38



Wykres 7 Wykres czasu wykonania algorytmu dla sposobów wyboru rozwiązania w sąsiedztwie



Wykres 8 Wykres zużycia pamięci dla różnych sposobów wyboru rozwiązania w sąsiedztwie



Wykres 9 Wykres średniej wielkości błędu dla różnych sposobów wyboru rozwiązania w sąsiedztwie

7. Analiza wyników i wnioski

Analiza wyników

Analiza wyboru różnych schematów chłodzenia

Z Wykresu 1 i Tabeli 1 można wyczytać, że średni czas wykonania algorytmu dla schematu chłodzenia geometrycznego i logarytmicznego (Boltzmanna) jest zbliżony dla instancji $n \leq 200$. Dla instancji n = 323 średni czas wykonania przy wykorzystaniu schematu chłodzenia logarytmicznego wzrasta 6-krotnie względem geometrycznego.
Średnia wielkość błędu dla instancji $n \leq 200$ jest zbliżona dla obu schematów. Dla instancji n = 323 wielkość błędu schematu geometrycznego jest około 30% większa od wielkości błędu schematu logarytmicznego.

Analiza wyboru różnych sposobów dobierania długości epoki

Z Wykresu 4 i Tabeli 2 można wyczytać, że dla sposobu dobierania długości epoki $n * 10$ wyniki generowane są kilkudziesięciokrotnie szybciej niż dla dobierania n^2. Wielkości błędu uzyskanego są minimalnie większe dla sposobu dobierania długości epoki $n * 10$ dla $n \leq 200$. Dla $n = 323$ wielkość błędu dla n^2 jest o około 50% mniejsza. Średnia wielkość błędu nie przekracza 30% dla instancji
n = <6, 200> w obu konfiguracjach.

Analiza wyboru różnych sposobów wyboru rozwiązania w sąsiedztwie

Sposób wyboru rozwiązania w sąsiedztwie oparty o wymianę łuków dla instancji $n \leq 200$ wykonuje się w czasie zbliżonym do sposobu 2-zamiany, lecz dla instancji n = 323 czas wykonania wymiany łuków jest dwukrotnie większy (wynosi około 10 sekund).
Średnia wielkość błędu uzyskanego rozwiązania jest o wiele większa
dla instancji n = <90, 200> dla sposobu 2-zamiany. Dla n = 323 średnia wielkość błędu wynosi 33%, gdy dla sposobu wymiany łuków średnia wielkość błędu wynosi 190%.
Dla sposobu wymiany łuków dla n = <6, 200> średnia wielkość błędu nie przekracza 30%.

Analiza zużycia pamięci

Z Tabel 1, 2 i 3 oraz Wykresów 2, 4 i 6 można wyczytać, że średnie zużycie pamięci dla wszystkich konfiguracji różni się maksymalnie około 10MiB i nie zmienia się względem wielkości instancji.

Wnioski

Dla metody SA zużycie pamięci jest niewielkie i wielkość instancji nie ma na to wpływu.
Dla prawie wszystkich instancji (poza rgb323.atsp) najszybsze i najbardziej zbliżone do wyniku optymalnego wyniki wygenerowała konfiguracja parametrów: schemat chłodzenia geometryczny, długość epoki = n*10, sposób dobierania sąsiedztwa metodą wymiany łuków.
Dla niektórych instancji problemu pewne konfiguracje generują wyniki bardziej zbliżone do optimum niż dla pozostałych. Dobór parametrów dla większości instancji powinien być podobny, lecz dla niektórych problemów (takich jak instancja rbg323.atsp) dobór parametrów powinien być spersonalizowany. Z uzyskanych wyników można zakładać, że dla instancji problemu rbg323.atsp aby uzyskać wynik jak najbardziej zbliżony do optimum należy wybrać schemat logarytmiczny i sposób doboru rozwiązania 2-zamiany.

8. Źródła

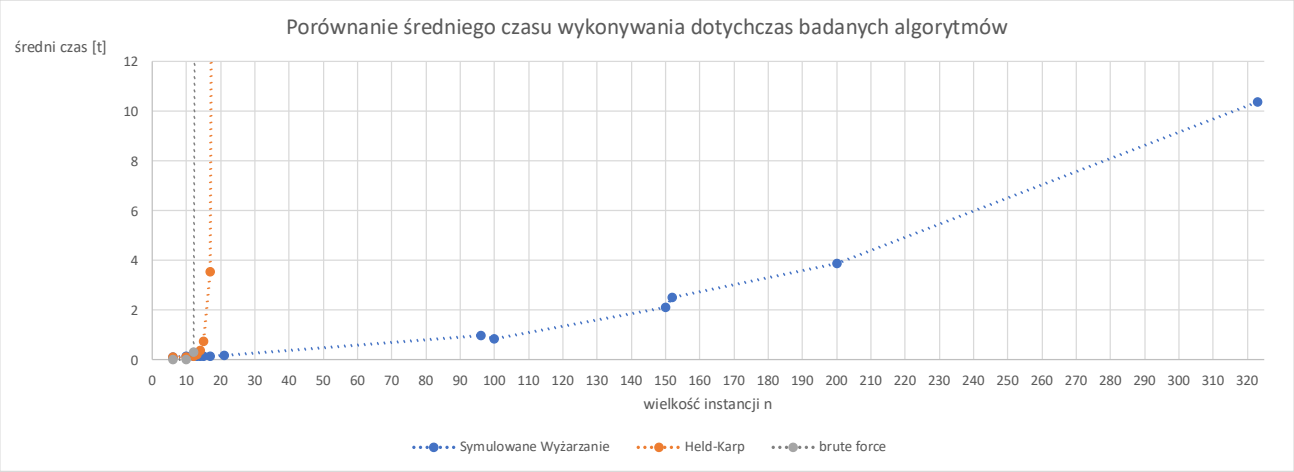
[1] J. Dreo, A. Petrowski, P. Siarry, E. Taillard – Metaheuristics for Hard Optimization
[2] https://docs.python.org/3/library/time.html
[3] https://pypi.org/project/memory-profiler/
[4] Sean Luke – Essentials of Metaheuristics
[5] El Ghazali Talbi – Metaheurstics. From design to implementation
[6]https://www.yumpu.com/xx/document/read/22987413/simulated-annealing-w-20-lat-paaaaniej-fatcat-agh
[7] https://pandas.pydata.org

Dodatek A

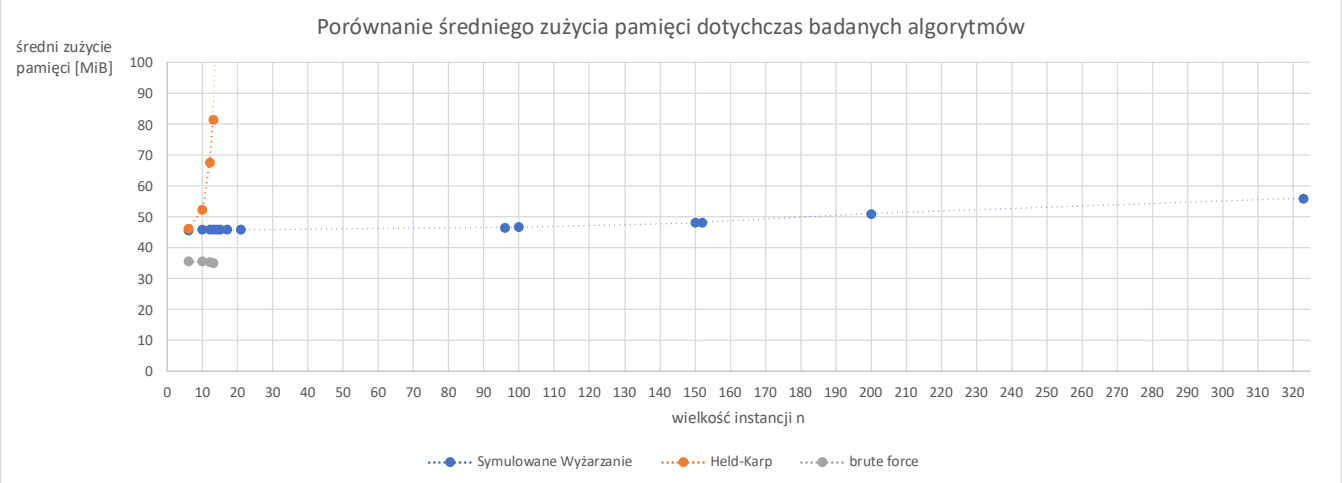
Porównanie algorytmów brute force, Helda-Karpa oraz symulowanego wyżarzania

Tabela A.1 Porównanie średnich wyników dotychczas badanych algorytmów

	ALGORYTM HELDA-KARPA		ALGORYTM BRUTE FORCE		Algorytm oparty o SA	
n	średni czas [s]	średnie zużycie pamięci [MiB]	średni czas [s]	średnie zużycie pamięci [MiB]	średni czas [s]	średnie zużycie pamięci [MiB]
6	0,1044	46,07	0,0001	35,61	0,1148	45,73
10	0,1263	52,34	0,0001	35,62	0,1530	45,82
12	0,1642	67,67	0,3232	35,37	0,1541	45,84
13	0,2273	81,46	40,3472	35,13	0,1567	45,86
14	0,3664	111,43	516,3495		0,1597	45,88
15	0,7400	193,95	7098,5659		0,1573	45,88
17	3,5391	561,01			0,1619	45,88
21	130,5848	2903,25			0,1719	45,88
96					0,9803	46,55
100					0,8415	46,71
150					2,1272	48,03
152					2,5040	48,20
200					3,8666	51,01
323					10,3837	55,95



Wykres A.1 Porównanie czasu wykonania dotychczas badanych algorytmów



Wykres A.2 Porównanie zużycia pamięci wykonania dotychczas badanych algorytmów

Analiza i wnioski

Algorytm oparty o metodę symulowanego wyżarzania ma złożoność pamięciową stałą, niezależną od wielkości instancji i pozwala wygenerować wyniki dla instancji $n < 20$ tak samo szybko jak algorytm Helda-Karpa oraz metoda siłowa. Metoda SA generuje wyniki dla instancji $n > 20$ w kilka sekund co jest nieosiągalne dla pozostałych dotychczas badanych algorytmów. Należy pamiętać, że metoda SA jest metaheurystyką i zwraca wyniki dążące do globalnego optimum, a nie dokładny, optymalny wynik.

W przypadku, gdy poszukuje się optimum dla dużych instancji ($n > 20$) lub najważniejsza jest szybkość generowania i wynik może mieć niewielki (do 30% przy dobrze dobranych parametrach) błąd względem wyniku optymalnego należy wybrać algorytm oparty o SA.

Dla instancji $n < 20$ i przy dużych zasobach pamięciowych algorytm Helda-Karpa będzie najlepszym wyborem, bo w krótkim czasie zwraca optymalny (dokładny) wynik. Przy ograniczonych zasobach pamięciowych i instancjach $n < 14$ pozostaje wybór metody siłowej.