Politechnika Wrocławska

Projektowanie efektywnych algorytmów

Prowadzący: Dr inż. Jarosław Mierzwa

Zadanie 2

Temat: Implementacja i analiza efektywności algorytmów **Tabu Search (TS)** i/lub **Symulowanego Wyżarzania (SW)** dla problemu komiwojażera (TSP).

Termin zajęć: PT 15:15 – 16:55

Autor sprawozdania: Karol Pastewski 252798

1. Wstęp teoretyczny

Rozpatrywanym problemem jest problem komiwojażera (*ang. travelling salesman problem, TSP*) w wersji asymetrycznej. Problem ten jest zagadnieniem optymalizacyjnym, polegającym na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Innymi słowami musimy znaleźć najkrótszą trasę przez wszystkie punkty, a następnie wrócić do początkowego punktu. W asymetrycznym TSP odległości między dwoma punktami mogą być różne. Do znalezienia najkrótszej ścieżki zastosowano dwa algorytmy: algorytm Tabu Search i algorytm Symulowanego Wyżarzania.

1.1. Algorytm Tabu Search

Algorytm Tabu Search jest algorytmem metaheurystycznym do rozwiazywania problemów optymalizacyjnych. Opiera się na iteracyjnym przeszukiwaniu przestrzeni rozwiązań wykorzystując sąsiedztwo elementów tej przestrzeni oraz zapamiętując przy tym przeszukiwaniu ostatnie ruchy, dopóki nie zostanie spełniony warunek końcowy. Ruchy są tymczasowo zapisywane do listy tabu, gdzie obecność danego ruchy oznacza zakaz wykonania go przez określona liczbę iteracji. Lista tabu ma za zadanie wyeliminować prawdopodobieństwo zapętleń przy przeszukiwaniu. Aby wykluczyć możliwość stagnacji można także dodać strategię dywersyfikacji, która pozwala na przeglądanie różnych obszarów przestrzeni rozwiązań.

Dla tej konkretnie implementacji algorytmu Tabu Search zastosowano tablicę tabu w formie kolejki, gdzie nowe ruchy są zapisywane na koniec kolejki, a gdy liczba elementów kolejki osiągnie wystarczający poziom, to pierwszy element kolejki jest usuwany.

Początkowe rozwiązanie jest znajdowane przy pomocy algorytmy najbliższego sąsiada. Czyli losowane jest początkowe miasto, od niego znajdowany jest najbliższy sąsiad, który nie został jeszcze rozpatrzony, tworzona jest krawędź między nimi, a następnie powtarzane są te kroki dla znalezionego sąsiada. Metoda ta kończy się, aż rozpatrzone zostaną wszystkie miasta.

Sąsiedztwo jest definiowane na trzy sposoby. W pierwszej definicji nowy sąsiad jest znajdowany zamieniając w ścieżce dwa losowe miasta. W drugiej losowane są trzy miasta i zmieniane są one w taki sposób: pierwszy z trzecim i drugi z trzecim. Ostatnia definicja polega na losowym znalezieniu dwóch miast i odwróceniu łuku, na który wskazuje zakres tych dwóch miast.

W tym programie zastosowano strategię dywersyfikacji działającą w taki sposób, że jeżeli przez określoną w programie liczbę iteracji algorytm nie znalazł lepszego rozwiązania, to następny nowy sąsiad jest kompletnie losowy.

Warunek końcowy jest określany przez użytkownika i jest to limit czasowy określony w sekundach.

1.2. Algorytm Symulowanego Wyżarzania (ang. Simulated Annealing)

Algorytm Symulowanego Wyżarzania jest także jest algorytmem metaheurystycznym. Ten algorytm także przeszukuje przestrzeń rozwiązań wykorzystując sąsiednie elementy, dopóki nie zostanie spełniony warunek stopu. Różnica jest taka, że SW wprowadza parametr temperatury do algorytmu i, w trakcie wykonywania się, zmienia ten parametr. Temperatura jest potrzebna do wychodzenia z minimów lokalnych, bo im większa temperatura, tym jest większe prawdopodobieństwo, że algorytm rozpatrzy gorsze sąsiednie rozwiązanie. Sprowadza się do tego, że na początku algorytmu, gdy temperatura jest duża, algorytm mocno skacze po rozwiązaniach w przestrzeni, ale w miarę ochładzania się temperatury, algorytm przestaje wybierać gorsze rozwiązania i stara się ulepszyć to, które już znalazł.

Parametrami używanymi w tym algorytmie oprócz temperatury są też długość epoki (liczba wewnętrznych iteracji algorytmu), funkcja zmiany temperatury i kryterium zatrzymania. Użyta funkcja zmiany temperatury jest funkcją geometryczną, czyli temperatura jest mnożona przez określony przez użytkownika współczynnik a (w sprawozdaniu a przyjmuje wartości 0.9, 0.95 i 0.99). Warunek stopu jest określany przez użytkownika i jest to limit czasowy w sekundach.

Początkowe rozwiązanie jest znajdowane w taki sam sposób jak w algorytmie Tabu Search, czyli algorytmem najbliższego sąsiada, a sąsiad jest znajdowany przy pomocy pierwszej definicji sąsiedztwa z Tabu Search, czyli zamiana dwóch miast.

2. Opis najważniejszych klas w projekcie

2.1. Tabu Search

```
63  /*
64   * Funkcja znajdująca sąsiada podanej w parametrze ścieżki.
65   * Sąsiad jest znajdowany zmieniając 2 losowe węzły miejscami ze
66   * sobą.
67   */
68  tuple<int, int> TabuSearch::randomNeighbor1(int *path) {
69      mt19937 gen(rd());
70      uniform_int_distribution<int> distribution(0, N - 1);
71      auto random = bind(distribution, gen);
72
73      int random1, random2;
74      random1 = random();
75
76      // Pętla zapobiega wylosowaniu się dwóch tych samych węzłów.
77      do {
78          random2 = random();
79      } while (random1 == random2);
80      swap(path[random1], path[random2]);
81      return {random1, random2};
82  }
```

Rysunek 1 Definicja sąsiedztwa: 2-zamiana

```
84  /*
85   * Funkcja znajdująca sąsiada podanej w parametrze ścieżki.
86   * Sąsiad jest znajdowany zmieniając 3 losowe węzły miejscami ze
87   * sobą.
88   */
89  tuple<int, int, int> TabuSearch::randomNeighbor2(int *path) {
90      mt19937 gen(rd());
91      uniform_int_distribution<int> distribution(0, N - 1);
92      auto random = bind(distribution, gen);
93
94      int random1, random2, random3;
95      random1 = random();
96
97      // Pętla zapobiega wylosowaniu się dwóch tych samych węzłów.
98      do {
99          random2 = random();
100     } while (random1 == random2);
101
102     // Pętla zapobiega wylosowaniu się dwóch tych samych węzłów.
103     do {
104         random3 = random();
105     } while (random1 == random3 || random2 == random3);
106     swap(path[random1], path[random3]);
107     swap(path[random2], path[random3]);
108     return {random1, random2, random3};
109 }
```

Rysunek 2 Definicja sąsiedztwa: 3-zamiana

```
111  /*
112   * Funkcja znajdująca sąsiada podanej w parametrze ścieżki.
113   * Sąsiad jest znajdowany znajdując 2 losowe węzły i odwracane są
114   * węzły na łuku wskazanym przez te dwa wylosowane węzły.
115   */
116  tuple<int, int> TabuSearch::randomNeighbor3(int *path) {
117      mt19937 gen(rd());
118      uniform_int_distribution<int> distribution(0, N - 1);
119      auto random = bind(distribution, gen);
120
121      int random1, random2;
122      random1 = random();
123      do {
124          random2 = random();
125      } while (random1 == random2);
126
127      for (int i = 0; i < abs(random1 - random2 + 1) / 2; ++i) {
128          if (random1 < random2) {
129              swap(path[random1 + i], path[random2 - i]);
130          } else {
131              swap(path[random2 + i], path[random1 - i]);
132          }
133      }
134      return {random1, random2};
135  }
```

Rysunek 3 Definicja sąsiedztwa: Wymiana łuków

```
225      // Tablica tabu, maksymalna wielkość jest określona w konstruktorze
226      list<int> move;
227      list<list<int>> tabuList;
228      int count = 0;
229      boolean isInTabu;
230      int node1, node2, node3 = -1;
231
232      //tabuList.push_back(bestPath);
233      oldPath.cost = bestPath.cost;
234      copyPath(oldPath.path, newPath.path);
```

Tablica tabu jest listą dwuwymiarową przechowującą dla albo trzy miasta, co ma symbolizować jaki ruch został użyty do znalezienia sąsiada. Zmienna isInTabu będzie potrzebna do określenia, czy ruch jest już w tablicy tabu.

```
237     while ((read_QPC() - start) / frequency < stop) {
238         copyPath(oldPath.path, newPath.path);
239 
240         // Znajdowanie sąsiada zależnie od wybranej definicji
241         if (neighborDefinition == 1) {
242             tie(node1, node2) = randomNeighbor1(newPath.path);
243             move.push_back(node1);
244             move.push_back(node2);
245         } else if (neighborDefinition == 2) {
246             tie(node1, node2, node3) = randomNeighbor2(newPath.path);
247             move.push_back(node1);
248             move.push_back(node2);
249             move.push_back(node3);
250         } else {
251             tie(node1, node2) = randomNeighbor3(newPath.path);
252             move.push_back(node1);
253             move.push_back(node2);
254         }
255 
256         newPath.cost = calcPathCost(newPath.path);
257         isInTabu = false;
258 
```

Głowna pętla programu kończy się limitem czasowym. Zależnie od wybranej opcji definicji wywoływana jest inna metoda znajdowana sąsiada, a także inaczej jest dodawany ruch do tablicy tabu (przy 2-zamianie i wymianie łuków dodawane są dwa miasta, a w 3-zamianie dodawane są trzy).

```
259         // Pętla sprawdza, czy sąsiad jest już w tabeli tabu
260         for (list path: tabuList) {
261             if (neighborDefinition == 1 || neighborDefinition == 2) {
262                 if (find(path.begin(), path.end(), node1) != path.end() &&
263                     find(path.begin(), path.end(), node2) != path.end()) {
264 
265                     isInTabu = true;
266                     break;
267                 }
268             } else {
269                 if (find(path.begin(), path.end(), node1) != path.end() &&
270                     find(path.begin(), path.end(), node2) != path.end() &&
271                     find(path.begin(), path.end(), node3) != path.end()) {
272 
273                     isInTabu = true;
274                     break;
275                 }
276             }
277 
278         }
279         count++;
```

Pętla sprawdza, czy ruch, którego użyliśmy do znalezienia sąsiada, jest już w tabeli tabu. Jeżeli tak, to zmienna isInTabu jest zmieniana na prawdę.

```
280             if (!isInTabu) {
281                 tabuList.push_back(move);
282                 if (newPath.cost < oldPath.cost) {
283                     copyPath(newPath.path, oldPath.path);
284                     oldPath.cost = newPath.cost;
285                 }
286                 if (newPath.cost < bestPath.cost) {
287                     //cout << newPath.cost << "\n";
288                     copyPath(newPath.path, bestPath.path);
289                     bestPath.cost = newPath.cost;
290                     moment = (read_QPC() - start) * 1'000 / frequency;
291                     count = 0;
292                 }
293 
294                 // Jeżeli tabela Tabu jest pełna, to pierwszy ruch jest usuwany
295                 if (tabuList.size() > maxTabuListSize) {
296                     tabuList.erase(tabuList.begin());
297                 }
298             }
299             move.clear();
300 
301             // Dywersyfikacja
302             if (count >= N * (N - 1) * (N - 2) * (N - 3)) {
303                 randomPath(oldPath.path);
304                 oldPath.cost = calcPathCost(oldPath.path);
305                 copyPath(oldPath.path, newPath.path);
306             }
307         }
308         showResults(moment);
309     }
```

Jeżeli ruchu nie ma w tablicy tabu, to sprawdzamy czy sąsiad jest lepszy od minimum lokalnego (zmienna oldPath), a następnie czy jest lepszy od minimum globalnego (bestPath). Jeżeli tablica tabu jest pełna, to usuwany jest pierwszy element, czyli usuwany element będzie mógł być w następnej iteracji użyty do znalezienia nowego sąsiada. Dywersyfikacja losuje nam nowego sąsiada, gdy algorytm wejdzie w stan stagnacji.

2.2. Symulowanego Wyżarzania

```
120 /*
121  * Funkcja ustawiająca początkową wartość temperatury.
122  */
123 double SimulatedAnnealing::setStartTemperature(int cost) {
124     return alpha * cost;
125 }
```

Początkowa wartość temperatury jest obliczana mnożąc współczynnik a i koszt początkowego rozwiązania.

```
126 /*
127  * Funkcja obliczającą nową wartość temperatury.
128  */
129 void SimulatedAnnealing::calcNewTemperature() {
130     temperature *= alpha;
131 }
```

Nowa wartość temperatury jest obliczana mnożąc współczynnik a z starą temperaturą.

```
164     // Pętla główna algorytmu
165     while ((read_QPC() - start) / frequency < stop) {
166         for (int i = 0; i < N * 50; ++i) {
167             copyPath(oldPath, newPath);
168             randomNeighbor(newPath);
169             newCost = calcPathCost(newPath);
170 
171             // Jeżeli nowa ścieżka jest lepsza od globalnie najlepszej, to zmieniana
172             // jest najlepsza globalna ścieżka.
173             if (newCost < bestCost) {
174                 //cout << newCost << "\n";
175                 copyPath(newPath, bestPath);
176                 bestCost = newCost;
177                 moment = (read_QPC() - start) * 1'000 / frequency;
178             }
179 
180             int delta = newCost - oldCost;
181             if (delta < 0) {
182                 copyPath(newPath, oldPath);
183                 oldCost = newCost;
184             } else {
185                 double x = random();
186                 if (x < exp(-delta / temperature)) {
187                     copyPath(newPath, oldPath);
188                     oldCost = newCost;
189                 }
190             }
191         }
192         calcNewTemperature();
193     }
194     if (oldCost < bestCost) {
195         copyPath(oldPath, bestPath);
196         bestCost = oldCost;
197         moment = (read_QPC() - start) * 1'000 / frequency;
198     }
199     showResults(moment);
200 }
```

Głowna pętla algorytmu. Długość epoki jest ustawiona na N*50. Po wykonaniu pętli jest obliczana nowa temperatura. W pętli znajdowany jest nowy sąsiad, jeżeli jest on lepszy od minimum globalnego, to zmieniane jest globalne minimum. Następnie sprawdzane jest, czy nowa ścieżka jest lepsza od lokalnego minimum, jeżeli tak, to zamieniamy go. Jeżeli nie jest, to przy pomocy prawdopodobieństwa możliwe jest wybranie gorszego rozwiązania. Pętla główna kończy się na limicie czasowym.

3. Plan eksperymentu

Dla Tabu Search:

> Dla każdej definicji sąsiedztwa, każdego pliku i warunku stopu (limit czasowy: 1, 5, 15, 30 [s]) powtórzono algorytm 10 razy.

Dla Symulowanego Wyżarzania:

> Dla każdej wartości współczynnika a (0.9, 0.95, 0.99), każdego pliku i warunku stopu (limit czasowy: 1, 5, 15, 30 [s]) powtórzono algorytm 10 razy.

Optymalne wartości dla rozpatrywanych plików:

- ftv47.atsp – 1776
- ftv170.atsp – 2755
- rbg403.atsp – 2465

4. Wyniki eksperymentów

4.1.Tabu Search

Plik: ftv47.atsp

Definicja sąsiedztwa: 2-zamiana

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2546	18	43,36%
2	2608	6	46,85%
3	2118	25	19,26%
4	2241	3	26,18%
5	*2327*	*17*	*31,02%*
6	2267	25	27,65%
7	2284	20	28,60%
8	2231	62	25,62%
9	2418	73	36,15%
10	2439	65	37,33%
Średnia:	**2347,9**	**31,4**	**32,20%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2397	27	34,97%
2	2225	45	25,28%
3	2307	48	29,90%
4	2468	11	38,96%
5	2297	27	29,34%
6	2185	4	23,03%
7	2083	16	17,29%
8	2274	48	28,04%
9	2516	34	41,67%
10	2382	15	34,12%
Średnia:	**2313,4**	**27,5**	**30,26%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2247	37	26,52%
2	2262	46	27,36%
3	2167	8	22,02%
4	2270	39	27,82%
5	2332	25	31,31%
6	2199	13	23,82%
7	2382	12	34,12%
8	2407	16	35,53%
9	2257	22	27,08%
10	2225	68	25,28%
Średnia:	**2274,8**	**28,6**	**28,09%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2374	45	33,67%
2	2344	6	31,98%
3	2080	16	17,12%
4	2259	23	27,20%
5	2168	23	22,07%
6	2417	67	36,09%
7	2334	65	31,42%
8	2249	12	26,63%
9	2291	80	29,00%
10	2167	62	22,02%
Średnia:	**2268,3**	**39,9**	**27,72%**

Definicja sąsiedztwa: 3-zamiana

Czas: 1 [s]

Nr	Koszt	**Czas [ms]**	Błąd
1	2478	388	39,53%
2	2202	857	23,99%
3	2143	406	20,66%
4	2215	428	24,72%
5	2406	460	35,47%
6	2154	746	21,28%
7	2193	504	23,48%
8	2383	732	34,18%
9	2261	562	27,31%
10	2289	831	28,89%
Średnia:	**2272,4**	**591,4**	**27,95%**

Czas: 15 [s]

Nr	**Koszt**	**Czas [ms]**	**Błąd**
1	2285	916	28,66%
2	2251	1129	26,75%
3	2259	2218	27,20%
4	2088	2059	17,57%
5	2270	1672	27,82%
6	2232	317	25,68%
7	2174	456	22,41%
8	2169	1293	22,13%
9	2200	2128	23,87%
10	2279	4779	28,32%
Średnia:	**2220,7**	**1696,7**	**25,04%**

Czas: 5 [s]

Nr	**Koszt**	**Czas [ms]**	**Błąd**
1	2260	1980	27,25%
2	2074	1347	16,78%
3	2222	363	25,11%
4	2135	912	20,21%
5	2269	2316	27,76%
6	2048	1136	15,32%
7	2232	1098	25,68%
8	2283	987	28,55%
9	2103	1491	18,41%
10	2204	609	24,10%
Średnia:	**2183**	**1223,9**	**22,92%**

Czas: 30 [s]

Nr	**Koszt**	**Czas [ms]**	**Błąd**
1	2251	2240	26,75%
2	2060	1682	15,99%
3	2092	2811	17,79%
4	2209	1288	24,38%
5	2315	2060	30,35%
6	2254	2224	26,91%
7	2343	2044	31,93%
8	2263	3665	27,42%
9	1968	1945	10,81%
10	2125	4028	19,65%
Średnia:	**2188**	**2398,7**	**23,20%**

Definicja sąsiedztwa: Wymiana łuków

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2217	4	24,83%
2	2502	126	40,88%
3	2262	63	27,36%
4	2344	32	31,98%
5	2135	12	20,21%
6	2202	27	23,99%
7	2146	1	20,83%
8	2168	61	22,07%
9	2453	17	38,12%
10	2554	18	43,81%
Średnia:	**2298,3**	**36,1**	**29,41%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2502	35	40,88%
2	2388	26	34,46%
3	2587	12	45,66%
4	2325	39	30,91%
5	2286	0	28,72%
6	2475	31	39,36%
7	2294	17	29,17%
8	2492	9	40,32%
9	2313	0	30,24%
10	2286	0	28,72%
Średnia:	**2394,8**	**16,9**	**34,84%**

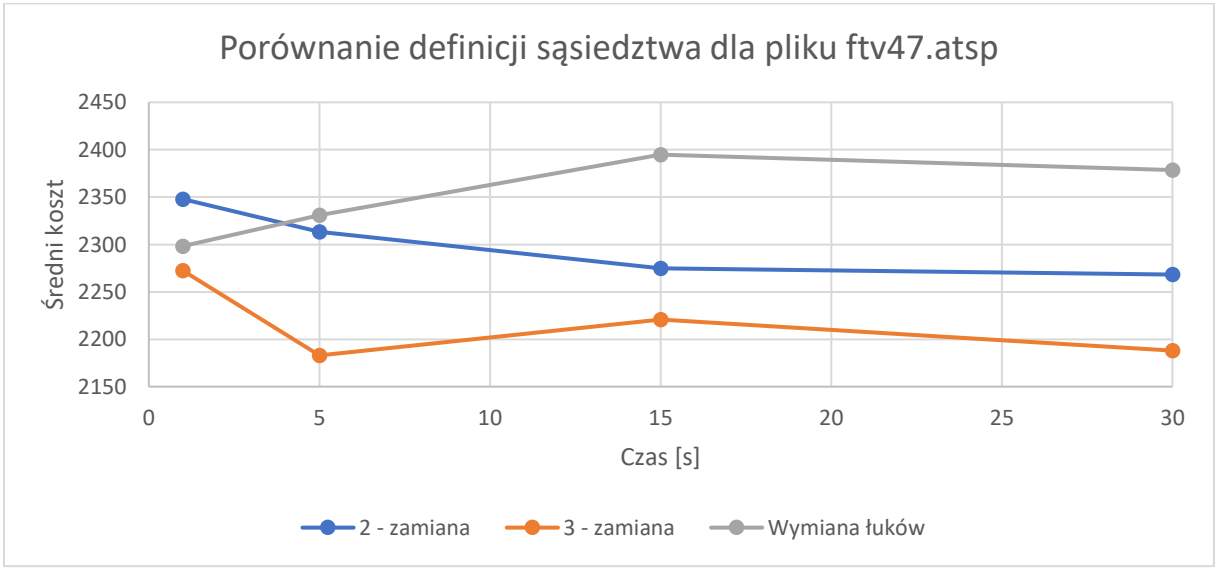
Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2317	34	30,46%
2	2517	73	41,72%
3	2272	28	27,93%
4	2286	0	28,72%
5	2168	64	22,07%
6	2492	16	40,32%
7	2408	36	35,59%
8	2225	68	25,28%
9	2285	40	28,66%
10	2339	12	31,70%
Średnia:	**2330,9**	**37,1**	**31,24%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2506	32	41,10%
2	2410	45	35,70%
3	2301	94	29,56%
4	2313	4	30,24%
5	2287	54	28,77%
6	2369	16	33,39%
7	2286	0	28,72%
8	2469	43	39,02%
9	2344	54	31,98%
10	2502	65	40,88%
Średnia:	**2378,7**	**40,7**	**33,94%**

Wykres:



Plik: ftv170.atsp

Definicja sąsiedztwa: 2-zamiana

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3654	757	32,63%
2	3631	647	31,80%
3	4043	672	46,75%
4	3861	666	40,15%
5	3749	705	36,08%
6	3845	945	39,56%
7	3909	942	41,89%
8	3870	867	40,47%
9	3956	768	43,59%
10	3924	944	42,43%
Średnia:	**3844,2**	**791,3**	**39,54%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3791	545	37,60%
2	3711	896	34,70%
3	3761	752	36,52%
4	3596	1580	30,53%
5	3795	551	37,75%
6	3782	1419	37,28%
7	3829	397	38,98%
8	3930	1167	42,65%
9	3895	565	41,38%
10	3728	2747	35,32%
Średnia:	**3781,8**	**1061,9**	**37,27%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3475	1527	26,13%
2	3805	1161	38,11%
3	3753	1066	36,23%
4	3769	1455	36,81%
5	3868	1165	40,40%
6	3753	1237	36,23%
7	3879	1162	40,80%
8	3668	1813	33,14%
9	3759	1174	36,44%
10	4083	1855	48,20%
Średnia:	**3781,2**	**1361,5**	**37,25%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3883	477	40,94%
2	3798	873	37,86%
3	3895	548	41,38%
4	3683	1534	33,68%
5	3721	938	35,06%
6	3559	1483	29,18%
7	3478	2940	26,24%
8	3829	1267	38,98%
9	3848	958	39,67%
10	3888	323	41,13%
Średnia:	**3758,2**	**1134,1**	**36,41%**

Definicja sąsiedztwa: 3-zamiana

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3965	0	43,92%
2	3850	387	39,75%
3	3788	0	37,50%
4	3939	0	42,98%
5	3814	70	38,44%
6	3942	0	43,09%
7	3847	974	39,64%
8	3931	657	42,69%
9	3985	796	44,65%
10	3925	0	42,47%
Średnia:	**3898,6**	**288,4**	**41,51%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	4040	4105	46,64%
2	3772	4093	36,91%
3	3919	579	42,25%
4	3888	2363	41,13%
5	3852	295	39,82%
6	3735	3354	35,57%
7	3942	0	43,09%
8	3869	3568	40,44%
9	3931	2605	42,69%
10	4015	4639	45,74%
Średnia:	**3896,3**	**2560,1**	**41,43%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3828	11433	38,95%
2	3665	12710	33,03%
3	3843	14992	39,49%
4	3884	8720	40,98%
5	3829	2571	38,98%
6	3977	8052	44,36%
7	4153	11509	50,74%
8	3816	10974	38,51%
9	3885	4710	41,02%
10	3829	14037	38,98%
Średnia:	**3870,9**	**9970,8**	**40,50%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3790	25420	37,57%
2	3871	25254	40,51%
3	3884	26114	40,98%
4	3954	18141	43,52%
5	3896	20679	41,42%
6	3867	21405	40,36%
7	3832	16177	39,09%
8	3775	17399	37,02%
9	3921	27610	42,32%
10	3848	29435	39,67%
Średnia:	**3863,8**	**22763,4**	**40,25%**

Definicja sąsiedztwa: Wymiana łuków

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3853	250	39,85%
2	3786	265	37,42%
3	4004	433	45,34%
4	3848	75	39,67%
5	3841	431	39,42%
6	3536	100	28,35%
7	3741	43	35,79%
8	3896	532	41,42%
9	4128	677	49,84%
10	3942	863	43,09%
Średnia:	**3857,5**	**366,9**	**40,02%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3781	760	37,24%
2	3922	1286	42,36%
3	3769	2362	36,81%
4	4058	2250	47,30%
5	4281	1920	55,39%
6	3861	54	40,15%
7	3961	3415	43,77%
8	3984	4690	44,61%
9	3901	158	41,60%
10	3935	3755	42,83%
Średnia:	**3945,3**	**2065**	**43,21%**

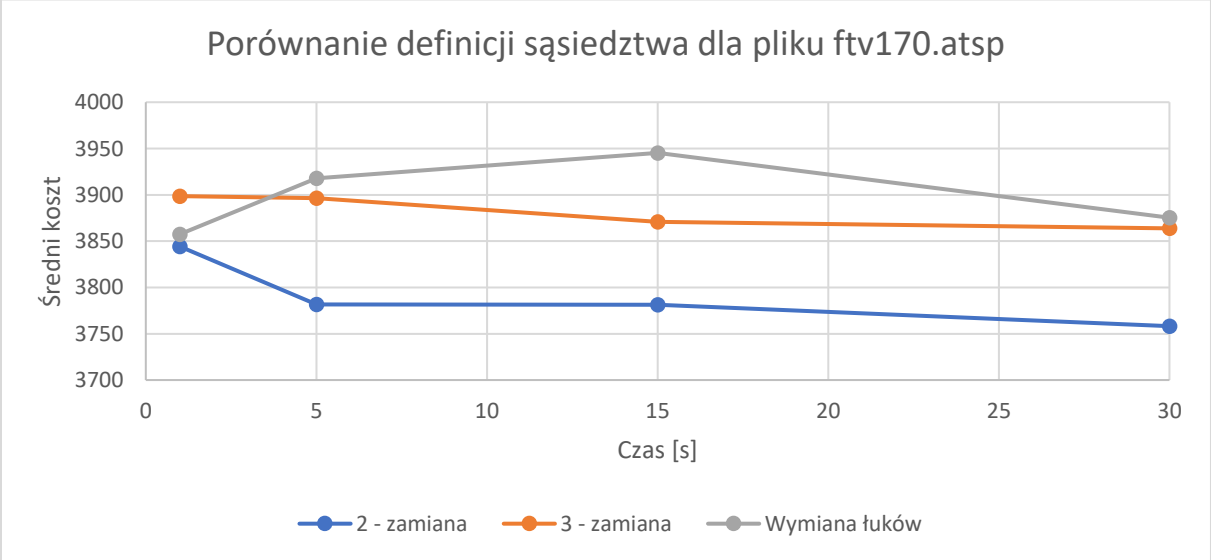
Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3721	991	35,06%
2	3885	150	41,02%
3	4140	3899	50,27%
4	3769	706	36,81%
5	3879	165	40,80%
6	4011	144	45,59%
7	3974	3119	44,25%
8	3798	157	37,86%
9	4023	0	46,03%
10	3977	481	44,36%
Średnia:	**3917,7**	**981,2**	**42,20%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3934	2173	42,79%
2	3934	2237	42,79%
3	3987	404	44,72%
4	3894	3643	41,34%
5	3854	3046	39,89%
6	3768	366	36,77%
7	3776	22	37,06%
8	3786	1547	37,42%
9	3861	94	40,15%
10	3958	619	43,67%
Średnia:	**3875,2**	**1415,1**	**40,66%**

Wykres:



Plik: rbg403.atsp

Definicja sąsiedztwa: 2-zamiana

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3034	967	23,08%
2	3089	994	25,31%
3	3109	998	26,13%
4	3042	979	23,41%
5	3109	984	26,13%
6	3045	939	23,53%
7	3042	978	23,41%
8	3080	995	24,95%
9	3085	974	25,15%
10	3094	993	25,52%
Średnia:	**3072,9**	**980,1**	**24,66%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2723	4935	10,47%
2	2752	4839	11,64%
3	2782	4921	12,86%
4	2781	4849	12,82%
5	2764	4890	12,13%
6	2750	4969	11,56%
7	2757	4926	11,85%
8	2741	4839	11,20%
9	2760	4772	11,97%
10	2794	4929	13,35%
Średnia:	**2760,4**	**4886,9**	**11,98%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2634	14850	6,86%
2	2629	13244	6,65%
3	2653	14879	7,63%
4	2591	14796	5,11%
5	2693	14921	9,25%
6	2594	14760	5,23%
7	2687	14218	9,01%
8	2666	14974	8,15%
9	2637	14911	6,98%
10	2656	14156	7,75%
Średnia:	**2644**	**14570,9**	**7,26%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2639	29145	7,06%
2	2620	22873	6,29%
3	2622	29906	6,37%
4	2592	29525	5,15%
5	2605	19671	5,68%
6	2632	22712	6,77%
7	2614	27808	6,04%
8	2621	28725	6,33%
9	2643	26877	7,22%
10	2671	28856	8,36%
Średnia:	**2625,9**	**26609,8**	**6,53%**

Definicja sąsiedztwa: 3-zamiana

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3245	986	31,64%
2	3340	981	35,50%
3	3322	991	34,77%
4	3283	945	33,18%
5	3289	989	33,43%
6	3262	914	32,33%
7	3326	972	34,93%
8	3262	948	32,33%
9	3217	991	30,51%
10	3271	930	32,70%
Średnia:	**3281,7**	**964,7**	**33,13%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2802	14261	13,67%
2	2855	14899	15,82%
3	2852	14962	15,70%
4	2866	14816	16,27%
5	2868	14979	16,35%
6	2833	14953	14,93%
7	2841	14409	15,25%
8	2854	14118	15,78%
9	2864	14823	16,19%
10	2845	14857	15,42%
Średnia:	**2848**	**14707,7**	**15,54%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2990	4999	21,30%
2	3030	4964	22,92%
3	3021	4862	22,56%
4	2998	4843	21,62%
5	3000	4986	21,70%
6	3103	4912	25,88%
7	3008	4855	22,03%
8	3028	4824	22,84%
9	3015	4963	22,31%
10	3027	4776	22,80%
Średnia:	**3022**	**4898,4**	**22,60%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2733	29550	10,87%
2	2746	29391	11,40%
3	2753	29984	11,68%
4	2743	28690	11,28%
5	2768	27981	12,29%
6	2800	29644	13,59%
7	2802	29253	13,67%
8	2798	29595	13,51%
9	2770	29938	12,37%
10	2769	29694	12,33%
Średnia:	**2768,2**	**29372**	**12,30%**

Definicja sąsiedztwa: Wymiana łuków

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3534	808	43,37%
2	3520	520	42,80%
3	3493	744	41,70%
4	3528	679	43,12%
5	3530	492	43,20%
6	3524	905	42,96%
7	3473	684	40,89%
8	3496	395	41,83%
9	3493	648	41,70%
10	3543	285	43,73%
Średnia:	**3513,4**	**616**	**42,53%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3465	7579	40,57%
2	3472	9518	40,85%
3	3422	11280	38,82%
4	3402	8498	38,01%
5	3481	7762	41,22%
6	3498	8907	41,91%
7	3378	12409	37,04%
8	3416	9835	38,58%
9	3470	14045	40,77%
10	3491	9439	41,62%
Średnia:	**3449,5**	**9927,2**	**39,94%**

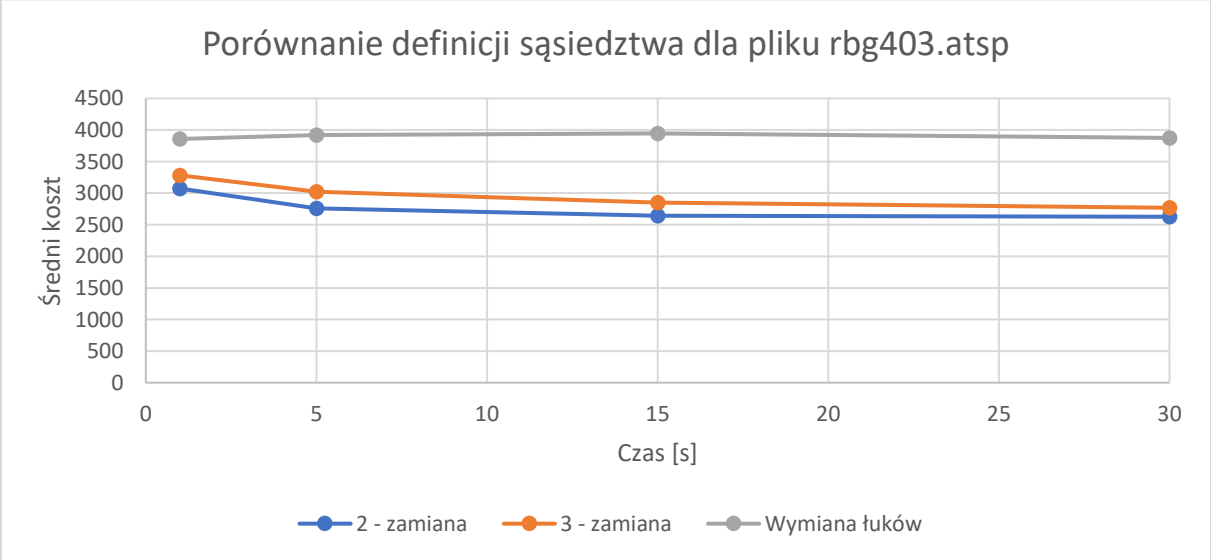
Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3471	4160	40,81%
2	3402	4337	38,01%
3	3498	159	41,91%
4	3445	2623	39,76%
5	3422	3165	38,82%
6	3515	4076	42,60%
7	3426	4678	38,99%
8	3398	4855	37,85%
9	3521	89	42,84%
10	3453	4935	40,08%
Średnia:	**3455,1**	**3307,7**	**40,17%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3498	14079	41,91%
2	3385	22347	37,32%
3	3480	29631	41,18%
4	3449	27718	39,92%
5	3453	27384	40,08%
6	3437	15192	39,43%
7	3420	11868	38,74%
8	3421	24664	38,78%
9	3421	9883	38,78%
10	3511	4285	42,43%
Średnia:	**3447,5**	**18705,1**	**39,86%**

Wykres:



4.2. Symulowane Wyżarzanie

Plik: ftv47.atsp

Współczynnik a: 0.9

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2133	307	20,10%
2	2025	274	14,02%
3	2240	250	26,13%
4	2220	323	25,00%
5	2062	307	16,10%
6	2135	302	20,21%
7	2062	264	16,10%
8	2200	297	23,87%
9	2103	274	18,41%
10	2267	249	27,65%
Średnia:	**2144,7**	**284,7**	**20,76%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2268	257	27,70%
2	2167	263	22,02%
3	2085	256	17,40%
4	2261	270	27,31%
5	2035	319	14,58%
6	2117	232	19,20%
7	2176	246	22,52%
8	2275	321	28,10%
9	2104	356	18,47%
10	2148	274	20,95%
Średnia:	**2163,6**	**279,4**	**21,82%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2236	247	25,90%
2	2213	277	24,61%
3	2057	267	15,82%
4	2039	229	14,81%
5	2192	293	23,42%
6	2134	320	20,16%
7	2186	309	23,09%
8	2198	249	23,76%
9	2168	279	22,07%
10	2258	261	27,14%
Średnia:	**2168,1**	**273,1**	**22,08%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2102	287	18,36%
2	2141	266	20,55%
3	2008	296	13,06%
4	2126	349	19,71%
5	2100	263	18,24%
6	2259	284	27,20%
7	2091	277	17,74%
8	2144	269	20,72%
9	2275	294	28,10%
10	2278	337	28,27%
Średnia:	**2152,4**	**292,2**	**21,19%**

Współczynnik a: 0.95

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	1942	514	9,35%
2	2139	503	20,44%
3	2024	523	13,96%
4	2154	511	21,28%
5	2010	566	13,18%
6	2083	514	17,29%
7	2034	511	14,53%
8	2085	546	17,40%
9	2140	476	20,50%
10	2101	491	18,30%
Średnia:	**2071,2**	**515,5**	**16,62%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2173	545	22,35%
2	2087	570	17,51%
3	2014	494	13,40%
4	2104	504	18,47%
5	2142	523	20,61%
6	2134	525	20,16%
7	1951	583	9,85%
8	2114	628	19,03%
9	2014	594	13,40%
10	1959	600	10,30%
Średnia:	**2069,2**	**556,6**	**16,51%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2063	743	16,16%
2	2093	541	17,85%
3	2121	554	19,43%
4	2137	493	20,33%
5	2173	515	22,35%
6	2240	496	26,13%
7	2024	515	13,96%
8	1970	524	10,92%
9	2000	573	12,61%
10	2156	589	21,40%
Średnia:	**2097,7**	**554,3**	**18,11%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2017	579	13,57%
2	1977	541	11,32%
3	2139	519	20,44%
4	2140	518	20,50%
5	2097	492	18,07%
6	1968	539	10,81%
7	2000	541	12,61%
8	2158	462	21,51%
9	2100	503	18,24%
10	2327	600	31,02%
Średnia:	**2092,3**	**529,4**	**17,81%**

Współczynnik a: 0.99

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2600	0	46,40%
2	2389	0	34,52%
3	2659	0	49,72%
4	2359	0	32,83%
5	2607	0	46,79%
6	2456	0	38,29%
7	2636	0	48,42%
8	2607	0	46,79%
9	2305	0	29,79%
10	2644	0	48,87%
Średnia:	**2526,2**	**0**	**42,24%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	1967	2618	10,75%
2	1920	2858	8,11%
3	2042	2550	14,98%
4	1850	2658	4,17%
5	2062	2416	16,10%
6	1898	2738	6,87%
7	1875	2525	5,57%
8	1979	2348	11,43%
9	1979	2722	11,43%
10	2131	2810	19,99%
Średnia:	**1970,3**	**2624,3**	**10,94%**

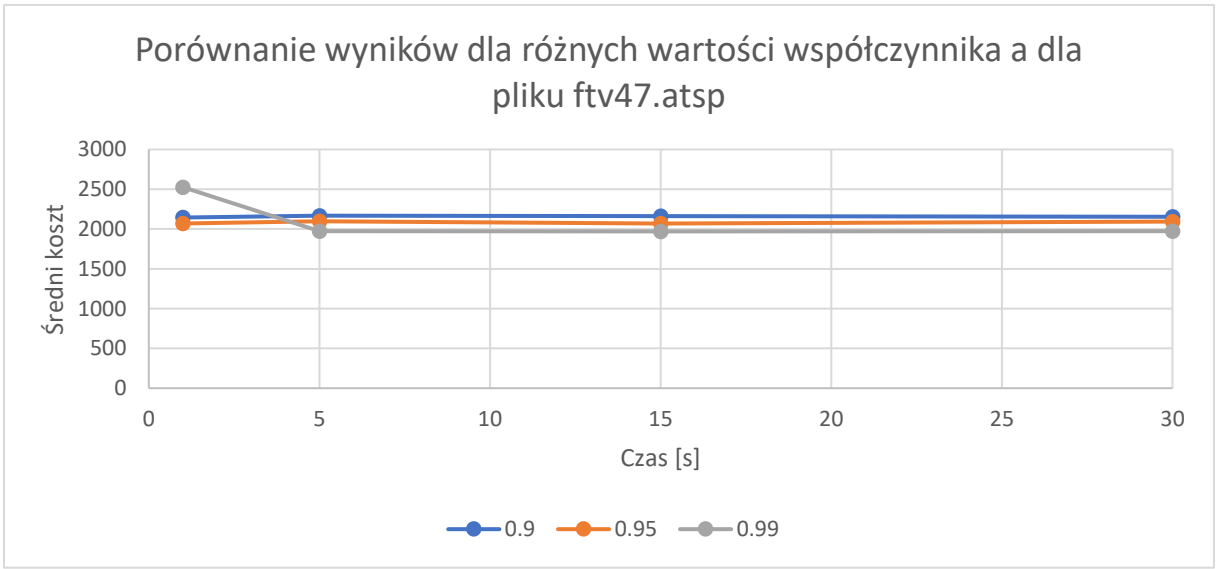
Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	1950	2821	9,80%
2	1946	2533	9,57%
3	1998	2404	12,50%
4	1907	2577	7,38%
5	1921	2448	8,16%
6	1916	2404	7,88%
7	2065	2739	16,27%
8	1955	2798	10,08%
9	2014	2586	13,40%
10	2056	2832	15,77%
Średnia:	**1972,8**	**2614,2**	**11,08%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	1881	2729	5,91%
2	1988	2556	11,94%
3	2057	2669	15,82%
4	1946	2486	9,57%
5	2002	2869	12,73%
6	1933	2753	8,84%
7	2015	2734	13,46%
8	2005	2419	12,89%
9	1982	2691	11,60%
10	1906	2674	7,32%
Średnia:	**1971,5**	**2658**	**11,01%**

Wykres:



Plik: ftv170.atsp

Współczynnik a: 0.90

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3942	0	43,09%
2	3887	0	41,09%
3	4170	0	51,36%
4	4081	0	48,13%
5	3975	0	44,28%
6	4041	0	46,68%
7	4000	0	45,19%
8	3965	0	43,92%
9	3966	0	43,96%
10	3942	0	43,09%
Średnia:	**3996,9**	**0**	**45,08%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	4057	0	47,26%
2	3925	0	42,47%
3	3942	0	43,09%
4	3999	0	45,15%
5	4081	0	48,13%
6	3991	0	44,86%
7	4029	0	46,24%
8	3982	0	44,54%
9	4146	0	50,49%
10	4157	0	50,89%
Średnia:	**4030,9**	**0**	**46,31%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	4029	0	46,24%
2	3804	0	38,08%
3	3783	0	37,31%
4	3965	0	43,92%
5	4082	0	48,17%
6	3837	0	39,27%
7	4083	0	48,20%
8	3952	0	43,45%
9	4111	0	49,22%
10	3981	0	44,50%
Średnia:	**3962,7**	**0**	**43,84%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	4039	0	46,61%
2	4047	0	46,90%
3	4015	0	45,74%
4	4083	0	48,20%
5	3949	0	43,34%
6	3991	0	44,86%
7	3949	0	43,34%
8	3939	0	42,98%
9	4111	0	49,22%
10	4148	0	50,56%
Średnia:	**4027,1**	**0**	**46,17%**

Współczynnik a: 0.95

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	4029	0	46,24%
2	4083	0	48,20%
3	4088	0	48,38%
4	3928	0	42,58%
5	3837	0	39,27%
6	3804	0	38,08%
7	3928	0	42,58%
8	4011	0	45,59%
9	3854	0	39,89%
10	4235	0	53,72%
Średnia:	**3979,7**	**0**	**44,45%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3966	0	43,96%
2	3824	0	38,80%
3	3999	0	45,15%
4	3830	0	39,02%
5	3965	0	43,92%
6	3975	0	44,28%
7	4047	0	46,90%
8	3931	0	42,69%
9	3788	0	37,50%
10	3810	0	38,29%
Średnia:	**3913,5**	**0**	**42,05%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3696	0	34,16%
2	3887	0	41,09%
3	3991	0	44,86%
4	3833	0	39,13%
5	4111	0	49,22%
6	3769	0	36,81%
7	4066	0	47,59%
8	4044	0	46,79%
9	3922	0	42,36%
10	4017	0	45,81%
Średnia:	**3933,6**	**0**	**42,78%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3965	0	43,92%
2	3919	0	42,25%
3	4082	0	48,17%
4	4064	0	47,51%
5	4071	0	47,77%
6	3920	0	42,29%
7	3854	0	39,89%
8	4007	0	45,44%
9	4071	0	47,77%
10	3920	0	42,29%
Średnia:	**3987,3**	**0**	**44,73%**

Współczynnik a: 0.99

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3837	0	39,27%
2	4044	0	46,79%
3	4143	0	50,38%
4	3854	0	39,89%
5	3698	0	34,23%
6	4081	0	48,13%
7	4066	0	47,59%
8	4066	0	47,59%
9	3824	0	38,80%
10	3894	0	41,34%
Średnia:	**3950,7**	**0**	**43,40%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3752	0	36,19%
2	3830	0	39,02%
3	3887	0	41,09%
4	3705	13104	34,48%
5	3812	13294	38,37%
6	3776	12763	37,06%
7	3802	0	38,00%
8	4111	0	49,22%
9	3919	0	42,25%
10	3939	0	42,98%
Średnia:	**3853,3**	**3916,1**	**39,87%**

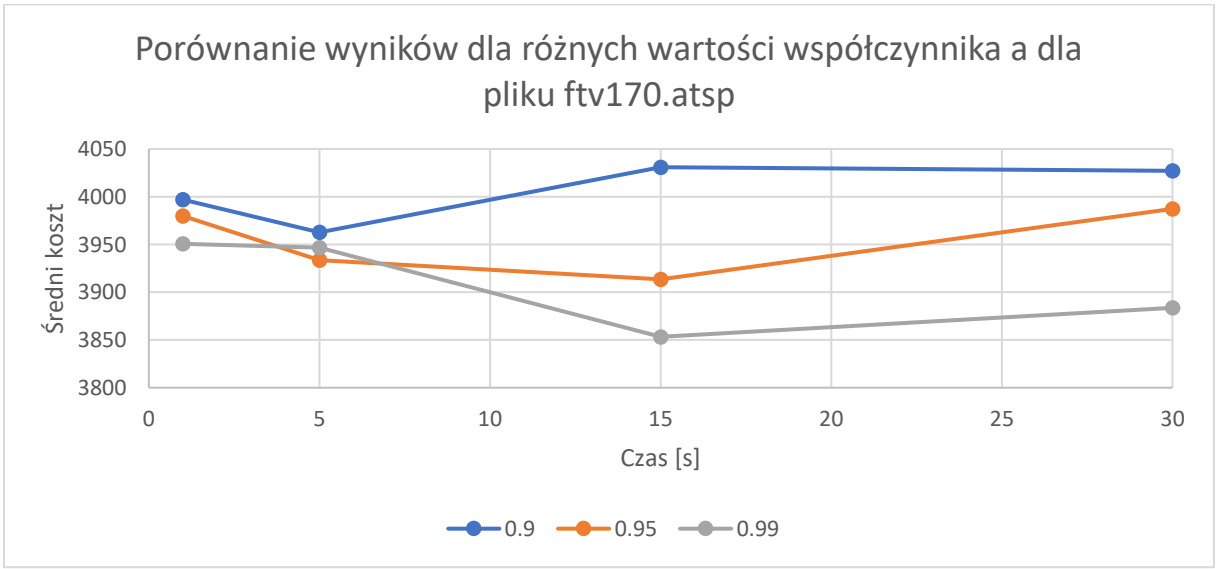
Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	4073	0	47,84%
2	3810	0	38,29%
3	3837	0	39,27%
4	3891	0	41,23%
5	3753	0	36,23%
6	4157	0	50,89%
7	4057	0	47,26%
8	4073	0	47,84%
9	4011	0	45,59%
10	3804	0	38,08%
Średnia:	**3946,6**	**0**	**43,25%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	4099	0	48,78%
2	3921	13718	42,32%
3	3974	0	44,25%
4	3860	13234	40,11%
5	3876	13053	40,69%
6	3925	0	42,47%
7	3582	0	30,02%
8	4017	14555	45,81%
9	3830	0	39,02%
10	3752	0	36,19%
Średnia:	**3883,6**	**5456**	**40,97%**

Wykres:



Plik: rbg403.atsp

Współczynnik a: 0.9

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3552	0	44,10%
2	3554	0	44,18%
3	3568	0	44,75%
4	3567	0	44,71%
5	3545	0	43,81%
6	3529	0	43,16%
7	3527	0	43,08%
8	3545	0	43,81%
9	3528	0	43,12%
10	3576	0	45,07%
Średnia:	**3549,1**	**0**	**43,98%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2537	13576	2,92%
2	2510	14298	1,83%
3	2559	13846	3,81%
4	2498	13676	1,34%
5	2521	14301	2,27%
6	2490	10499	1,01%
7	2547	14750	3,33%
8	2515	12665	2,03%
9	2521	11472	2,27%
10	2527	7951	2,52%
Średnia:	**2522,5**	**12703,4**	**2,33%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2623	4957	6,41%
2	2565	4946	4,06%
3	2574	4997	4,42%
4	2633	4920	6,82%
5	2554	5020	3,61%
6	2611	5041	5,92%
7	2583	4982	4,79%
8	2637	4983	6,98%
9	2574	4937	4,42%
10	2593	4981	5,19%
Średnia:	**2594,7**	**4976,4**	**5,26%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2496	26448	1,26%
2	2497	29155	1,30%
3	2517	20687	2,11%
4	2510	26214	1,83%
5	2531	21283	2,68%
6	2519	17914	2,19%
7	2505	28507	1,62%
8	2517	27623	2,11%
9	2515	24477	2,03%
10	2518	19206	2,15%
Średnia:	**2512,5**	**24151,4**	**1,93%**

Współczynnik a: 0.95

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3544	0	43,77%
2	3577	0	45,11%
3	3534	0	43,37%
4	3540	0	43,61%
5	3530	0	43,20%
6	3524	0	42,96%
7	3507	0	42,27%
8	3529	0	43,16%
9	3528	0	43,12%
10	3539	0	43,57%
Średnia:	**3535,2**	**0**	**43,42%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2494	14248	1,18%
2	2541	14942	3,08%
3	2539	14979	3,00%
4	2503	13864	1,54%
5	2508	11854	1,74%
6	2505	13688	1,62%
7	2539	14550	3,00%
8	2512	13325	1,91%
9	2519	13949	2,19%
10	2504	12511	1,58%
Średnia:	**2516,4**	**13791**	**2,09%**

Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3504	0	42,15%
2	3497	0	41,87%
3	3534	0	43,37%
4	3524	0	42,96%
5	3555	0	44,22%
6	3532	0	43,29%
7	3540	0	43,61%
8	3534	0	43,37%
9	3549	0	43,98%
10	3554	0	44,18%
Średnia:	**3532,3**	**0**	**43,30%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	2509	19380	1,78%
2	2495	14995	1,22%
3	2518	29872	2,15%
4	2531	23903	2,68%
5	2546	24351	3,29%
6	2493	19719	1,14%
7	2507	17669	1,70%
8	2500	27630	1,42%
9	2482	18041	0,69%
10	2516	28341	2,07%
Średnia:	**2509,7**	**22390,1**	**1,81%**

Współczynnik a: 0.99

Czas: 1 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3549	0	43,98%
2	3529	0	43,16%
3	3532	0	43,29%
4	3557	0	44,30%
5	3532	0	43,29%
6	3539	0	43,57%
7	3551	0	44,06%
8	3539	0	43,57%
9	3579	0	45,19%
10	3550	0	44,02%
Średnia:	**3545,7**	**0**	**43,84%**

Czas: 15 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3535	0	43,41%
2	3579	0	45,19%
3	3520	0	42,80%
4	3549	0	43,98%
5	3519	0	42,76%
6	3535	0	43,41%
7	3546	0	43,85%
8	3536	0	43,45%
9	3549	0	43,98%
10	3544	0	43,77%
Średnia:	**3541,2**	**0**	**43,66%**

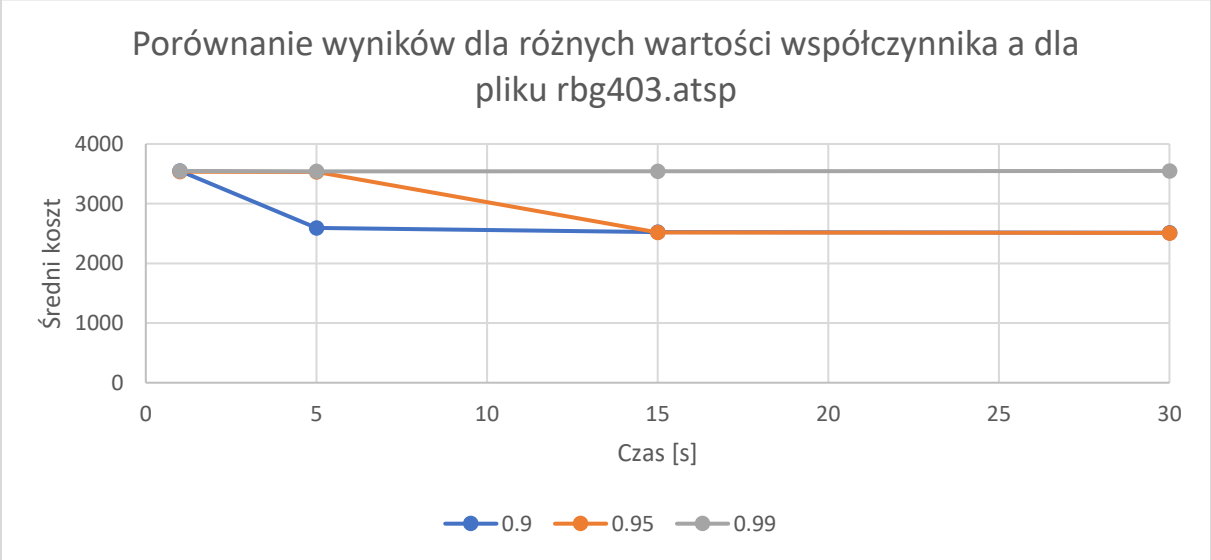
Czas: 5 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3567	0	44,71%
2	3545	0	43,81%
3	3534	0	43,37%
4	3571	0	44,87%
5	3543	0	43,73%
6	3545	0	43,81%
7	3507	0	42,27%
8	3524	0	42,96%
9	3540	0	43,61%
10	3539	0	43,57%
Średnia:	**3541,5**	**0**	**43,67%**

Czas: 30 [s]

Nr	Koszt	Czas [ms]	Błąd
1	3551	0	44,06%
2	3563	0	44,54%
3	3544	0	43,77%
4	3539	0	43,57%
5	3534	0	43,37%
6	3533	0	43,33%
7	3555	0	44,22%
8	3582	0	45,31%
9	3547	0	43,89%
10	3536	0	43,45%
Średnia:	**3548,4**	**0**	**43,95%**

Wykres:



Najlepsze znalezione ścieżki:

Tabu Search:

ftv47.atsp:

- Koszt: 1968
- Ścieżka: 14 16 45 39 21 20 37 38 0 25 40 47 26 1 11 18 13 34 23 12 32 7 31 24 4 29 30 5 6 10 8 9 33 3 27 28 2 43 42 22 41 19 44 15 17 46 36 35 14

ftv170.atsp:

- Koszt: 3475
- Ścieżka: 25 150 160 151 152 142 141 134 131 113 164 127 126 125 124 121 120 122 123 162 102 103 117 118 119 129 128 130 135 138 139 140 6 7 8 9 10 76 74 75 11 12 18 19 20 37 22 23 26 27 28 29 30 31 33 34 156 40 39 38 35 49 170 73 77 1 2 0 81 80 79 82 78 72 71 60 50 51 52 53 43 55 54 58 59 61 68 67 167 70 87 85 86 83 84 69 66 63 64 56 57 62 65 88 153 154 89 90 91 94 96 97 99 98 95 92 93 166 108 107 106 105 165 163 100 101 104 110 109 114 115 116 136 137 147 148 149 161 14 13 21 32 158 36 157 41 155 42 45 44 46 47 48 168 3 4 5 133 169 111 112 132 143 144 146 145 159 16 17 24 15 25

rbg403.atsp

- Koszt: 2592
- Ścieżka: 20 23 14 62 13 205 204 24 36 32 274 46 33 376 68 38 270 138 18 402 287 83 43 34 295 50 56 394 225 58 8 6 64 3 2 61 107 386 47 112 128 137 65 397 260 176 286 273 272 28 232 374 126 10 27 11 310 29 77 31 52 40 39 78 226 147 79 69 245 81 22 21 82 249 130 172 26 182 152 76 160 15 333 267 281 221 67 66 60 44 88 96 94 90 72 57 164 25 352 92 51 49 37 247 256 120 392 351 293 150 213 41 208 312 35 364 303 71 239 70 349 115 114 353 263 99 389 318 97 206 30 131 355 168 242 117 180 359 307 143 59 340 129 103 209 278 122 119 214 192 189 104 9 384 383 203 146 144 105 391 154 111 369 139 4 264 328 317 108 255 102 265 275 210 110 326 258 207 45 363 289 162 48 358 327 170 284 290 125 74 216 200 198 155 323 305 215 309 252 191 187 234 228 224 178 183 17 12 217 285 282 357 322 370 121 132 243 298 291 315 161 148 325 134 306 319 135 280 279 136 308 211 98 254 223 344 345 173 354 342 266 141 116 218 238 229 393 149 347 401 400 385 300 236 227 381 220 197 171 63 297 86 240 219 19 337 181 346 338 331 378 398 396 156 390 159 7 388 360 194 324 212 169 241 330 343 85 248 348 329 294 387 84 257 42 177 253 158 1 93 145 193 87 283 277 341 124 269 166 199 91 367 75 262 261 301 195 201 118 174 250 202 190 100 186 53 73 366 321 379 109 375 371 372 356 127 268 16 246 95 271 350 304 5 222 185 231 288 251 314 235 113 395 332 296 233 380 362 244 80 230 382 320 184 151 101 336 292 153 368 142 276 259 188 299 302 89 311 55 106 165 373 140 334 196 0 335 163 365 133 361 175 179 316 157 339 237 123 399 377 167 54 313 20

Symulowane Wyżarzanie:

ftv47.atsp

- Koszt: 1875
- Ścieżka: 22 19 44 15 45 39 21 40 47 26 1 10 11 0 25 20 37 38 18 46 36 35 14 16 17 13 34 23 12 32 7 31 5 4 3 6 24 29 30 8 9 33 27 28 2 41 43 42 22

ftv170.atsp

- Koszt: 3705
- Ścieżka: 2 1 0 112 132 110 109 107 106 105 97 165 164 127 126 125 124 129 136 128 130 131 163 98 95 96 104 114 113 115 116 117 118 119 120 102 103 108 166 154 89 90 91 87 70 69 67 66 65 88 153 83 84 71 50 49 170 73 168 72 60 63 64 56 55 58 59 51 52 53 43 44 45 46 47 78 82 79 80 81 5 133 169 111 3 4 9 10 76 74 75 11 12 158 31 54 57 62 61 68 167 85 86 93 92 94 99 100 101 123 162 122 121 146 145 137 135 138 139 140 141 134 6 7 8 15 159 16 17 18 19 20 21 29 30 28 27 26 23 22 24 25 150 160 152 142 144 143 147 148 149 161 151 14 13 32 36 157 33 34 35 37 38 39 40 156 155 41 42 48 77 2

rbg403.atsp

- Koszt: 2482
- Ścieżka: 196 0 64 15 253 42 130 201 118 176 156 325 129 255 283 8 400 361 146 320 50 209 41 82 247 373 202 23 14 62 13 306 344 111 369 33 376 160 365 237 162 48 135 280 279 188 173 319 38 270 254 223 6 102 123 166 67 114 69 163 3 91 90 72 198 312 35 402 287 78 382 381 128 153 75 251 274 281 139 175 374 363 379 242 117 193 136 333 267 107 386 2 61 46 125 60 98 99 389 286 273 357 358 293 137 110 207 45 218 145 126 271 327 151 101 235 121 47 112 87 56 195 183 17 12 184 4 265 275 154 217 9 304 394 393 211 131 355 52 40 263 206 30 68 32 113 220 197 232 164 25 158 1 316 351 190 138 18 301 120 116 161 148 210 133 86 11 152 310 115 338 106 165 375 371 257 124 339 24 54 172 26 387 384 383 203 95 55 352 169 27 336 292 343 397 5 150 388 348 231 205 204 372 28 340 180 359 314 59 65 285 282 83 248 332 296 159 323 309 181 346 66 74 29 268 16 262 261 245 174 185 246 233 324 212 134 288 249 234 228 71 149 51 298 291 315 302 89 399 122 401 97 276 259 182 178 100 44 141 392 256 240 219 391 390 85 311 10 79 96 189 236 227 144 170 284 290 366 321 289 208 239 70 177 103 109 326 147 77 31 168 167 241 330 329 294 317 367 76 43 34 308 260 104 349 214 192 94 132 243 238 229 225 105 19 57 142 244 80 335 230 226 258 385 199 295 313 7 140 334 157 269 88 353 39 345 299 49 215 73 194 380 362 360 354 200 398 396 370 322 350 331 378 377 171 63 224 81 22 21 364 303 337 186 53 305 37 36 318 179 368 213 356 58 277 341 155 143 119 216 342 297 252 191 187 222 250 221 300 307 20 108 93 84 395 127 92 347 264 328 272 278 266 196

5. Wnioski

W Tabu Search widzimy, że wybór odpowiedniej definicji sąsiedztwa może być kluczowy. W średnim i dużym pliku najlepiej sprawowała się 2-zamiana, a w małym pliku 3-zamiana. W każdym pliku najgorzej wypadała wymiana łuków. Do każdego problemu powinno się podchodzić indywidualnie i dobierać takie parametry, które dają najlepszy wynik w konkretnym zadaniu.

W Symulowanym Wyżarzaniu widzimy, że im większy współczynnik a, tym dokładniejsze są wyniki, ale potrzeba więcej czasu, aby do nich dojść. Dobrze to widać na przykładzie dużego pliku, gdzie przy współczynniku a równym 0.99 wyniki były gorsze niż przy niższych wartościach przez to, że temperatura nie zdążyła się odpowiednio szybko schłodzić. Zaproponowana implementacja Symulowanego Wyżarzania ma problemy z średnim plikiem, gdzie nie może znaleźć lepszego rozwiązania, niż to początkowe. Należało by zapewne dostosować lepiej parametry pod ten typ problemu.

Oprócz wyjątku, jakim jest plik średni algorytm Symulowanego Wyżarzania jest bliżej optymalnego rozwiązania od algorytmu Tabu Search. W dużym pliku nawet znalazł rozwiązanie o błędzie względnym wynoszącym poniżej 1%. Niestety, tak jak było to wspomniane, Symulowane Wyżarzanie nie poradziło sobie z średnim plikiem, więc można stwierdzić, że Tabu Search jest bardziej uniwersalnym algorytmem.

6. Źródła

- https://www.ii.uni.wroc.pl/~prz/2011lato/ah/opracowania/szuk_tabu.opr.pdf
- https://cs.pwr.edu.pl/zielinski/lectures/om/localsearch.pdf
- http://www2.imm.dtu.dk/courses/02719/tabu/4tabu2.pdf
- http://www.pi.zarz.agh.edu.pl/intObl/notes/IntObl_w2.pdf
- http://155.158.112.25/~algorytmyewolucyjne/materialy/algorytm_symulowanego_wyzarzania.pdf
- https://en.wikipedia.org/wiki/Tabu_search
- http://www.cs.put.poznan.pl/mhapke/TO-LS.pdf