Projektowanie Efektywnych Algorytmów

Zadanie projektowe nr 2

Implementacja i analiza efektywności algorytmu Tabu Search i Symulowanego Wyżarzania dla problemu komiwojażera

Piątek, 11:15

Bartosz Szymczak
252 734

prowadzący
Dr inż. Jarosław Mierzwa

Informatyka Techniczna
Wydział Informatyki i Telekomunikacji

4 stycznia 2022

Spis treści

1 Wstęp teoretyczny

1.1 Podstawowe założenia

Podczas realizacji zadania należy przyjąć następujące założenia:

- używane struktury danych powinny być alokowane dynamicznie,
- program powinien umożliwić wczytanie danych testowych z plików: ftv47.atsp, ftv170.atsp, rgb403.atsp ze strony http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/atsp/,
- program musi umożliwiać wprowadzenia kryterium stopu jako czasu wykonania podawanego w sekundach,
- implementacje algorytmów należy dokonać zgodnie z obiektowym paradygmatem programowania,
- kod źródłowy powinien być komentowany.

1.2 Opis algorytmów

1.2.1 Tabu Search

Przeszukiwanie tabu jest sposobem rozwiązywania problemów optymalizacyjnych, opartą na iteracyjnym przeszukiwaniu przestrzeni rozwiązań, wykorzystując sąsiedztwo pewnych elementów tej przestrzeni oraz zapamiętując przy tym przeszukaniu ostatnie ruchy, dopóki nie zostanie spełniony warunek końcowy. Obecność danego ruchu na liście tabu jest tymczasowa i oznacza, że danego ruchu nie można wykonać przez określoną liczbę iteracji. Lista tabu ma za zadanie zmniejszyć prawdopodobieństwo zapętleń przy przeszukiwaniu i zmusić algorytm do dywersyfikacji, czyli przeszukania nowych, niezbadanych rejonów przestrzeni przeszukiwań. Sąsiedztwo elementów to relacja na parach elementów przestrzeni, która obejmuje dziedzinę przestrzeni przeszukiwań. Często wykorzystuje się dodatkowo kryterium aspiracji, które pozwala na wykonanie ruchu tabu, w celu uniknięcia zapętleń lub braku możliwości wykonania ruchu.

Dla problemu komiwojażera przestrzenią rozwiązań są poszczególne miasta i trasy pomiędzy nimi, rozwiązaniem jest ścieżka od pierwszego miasta, przez wszystkie podane aż do ponownego odwiedzenia pierwszego oraz jej koszt. Sąsiedztwem są pozycje miast w stworzonej ścieżce, a lista tabu zawierać będzie indeksy miast o zmodyfikowanej pozycji oraz "czas życia" elementu. Pierwsze rozwiązanie zostanie wyznaczone prostym algorytmem zachłannym.

1.2.2 Symulowane wyżarzanie

Algorytm symulowanego wyżarzania jest rozwinięciem iteracyjnych metod optymalizacyjnych. Istotną różnicą jest możliwość wyboru gorszego rozwiązania. Taki wybór dokonywany jest z pewnym prawdopodobieństwem. Dzięki temu algorytm może w określonych warunkach wyjść z lokalnej grupy rozwiązań i dotrzeć do globalnego rozwiązania. Parametrem, który ma wpływ na prawdopodobieństwo wyboru gorszego rozwiązania jest "temperatura". Im wyższa, tym prawdopodobieństwo wyboru gorszego rozwiązania jest większe. Na początku, temperatura jest wysoka, dzięki czemu algorytm może często zmieniać rozwiązania, niejednokrotnie wybierając gorsze rozwiązanie. Wraz z kolejnymi iteracjami algorytmu temperatura spada i częściej wybierane są rozwiązania lepsze.

2 Implementacja poszczególnych algorytmów

2.1 Tabu Search

Struktura elementów tabu TabuSearch przechowuje informacje o indeksach modyfikowanych miast oraz o "czasie życia" elementu.

Za metodę przeszukiwania tabu odpowiedzialna jest klasa TabuSearch. Jej istotnymi zmiennymi są:

- vector<TabuElement> tabuList - wektor przechowujący elementy tabu,
- vector<vector<int>> temp - macierz miast i tras pomiędzy nimi, nad którą będą przeprowadzane obliczenia,
- int size - ilość miast w podanej macierzy,
- bool diversification - zmienna przechowująca istnienie dywersyfikacji,
- float stopCriterion - zmienna przechowująca wartość kryterium stopu,
- int firstTown - zmienna przechowująca indeks pierwszego miasta,

Klasa składa się z przeciążonego konstruktora, jednej głównej metody oraz pięciu metod pomocniczych.

- Konstruktor przypisuje początkowe wartości zmiennym, przekazywane argumenty to macierz miast i tras pomiędzy nimi, nad którą będą przeprowadzane obliczenia oraz kryterium stopu. Miasto początkowe zostaje ustawione na miasto o indeksie 0.
- Metoda główna doTabuSearch wykonuje algorytm przeszukiwania tabu nad dostarczonymi w konstruktorze danymi. Metoda początkowo tworzy rozwiązanie przy użyciu metody pomocniczej greedyTS, następnie ustawia właściwości dywersyfikacji. Dalej wykonuje się pętla przez czas określony jako kryterium stopu. W niej początkowo wyznaczany jest kolejny element listy tabu oraz modyfikowane jest tymczasowe rozwiązanie. Dalej w przypadku rozpoznania lepszego rozwiązania od poprzednich, dywersyfikacja zostaje zresetowana oraz zapisywana jest nowa najlepsza ścieżka. Następnie sprawdzana jest aktualna wartość skojarzona z dywersyfikacją, jeśli jest nie większa od zera to zostaje stworzone losowe rozwiązanie oraz zresetowana zostaje właściwość dywersyfikacji. Dodatkowo sprawdzane jest, czy nowe rozwiązanie jest lepsze od poprzednich. Na koniec pętli właściwość dywersyfikacji zostaje zredukowana. Metoda ostatecznie wypisuje wyznaczoną ścieżkę wraz z czasem wyznaczenia oraz kosztem.
- Metoda pomocnicza greedyTS odpowiedzialna jest za wyznaczenie ścieżki, przy użyciu prostego algorytmu zachłannego. Algorytm ten polega na szukaniu najkrótszych ścieżek wychodzących z rozpatrywanego miasta. Następnie brane pod uwagę są miasta do których prowadzą wyznaczone poprzednio ścieżki. Metoda modyfikuje wektor ścieżki przekazany w argumencie oraz zwraca koszt wyznaczonej ścieżki.
- Metoda pomocnicza localSolution odpowiedzialna jest za przeszukiwanie lokalnych przestrzeni w celu wyznaczenia najlepszego rozwiązania. Miasta obecnie rozpatrywanej ścieżki są zamieniane pozycjami, dzięki pomocniczej metodzie swap, a następnie sprawdzane są wartości nowo powstałych rozwiązań. W przypadku znalezienia lepszej ścieżki, algorytm tworzy element listy tabu i dodaje go do wektora listy tabu, jeśli posiada jeszcze ustalone miejsce. Dodatkowo aktualizowany jest "czas życia" każdego elementu.

- Metoda pomocnicza swap zamienia miejscami miasta o podanych indeksach w podanej ścieżce.
- Metoda pomocnicza inTabuList sprawdza, czy element o podanych indeksach miast znajduje się w liście tabu.
- Metoda pomocnicza countPath oblicza i zwraca koszt podanej ścieżki.

2.2 Symulowane wyżarzanie

Klasa SimulatedAnealing, która odpowiedzialna jest za przeprowadzanie symulowanego wyżarzania dzieli część metod z klasą TabuSearch. Główne zmiany zachodzą w metodach doSimulatedAnealing oraz insert. Jej istotnymi zmiennymi są:

- float temperature - zmienna przechowująca wartość temperatury,
- float minTemperature - zmienna przechowująca minimalną wartość temperatury,
- vector<vector<int>> temp - macierz miast i tras pomiędzy nimi, nad którą będą przeprowadzane obliczenia,
- int size - ilość miast w podanej macierzy,
- float stopCriterion - zmienna przechowująca wartość kryterium stopu,
- int firstTown - zmienna przechowująca indeks pierwszego miasta,

Klasa składa się z przeciążonego konstruktora, jednej głównej metody oraz trzech metod pomocniczych.

- Konstruktor przypisuje początkowe wartości zmiennym, przekazywane argumenty to macierz miast i tras pomiędzy nimi, nad którą będą przeprowadzane obliczenia oraz kryterium stopu. Miasto początkowe zostaje ustawione na miasto o indeksie 0,
- Metoda główna doSimulatedAnealing wykonuje algorytm symulowanego wyżarzania na danych podanych w konstruktorze. Metoda początkowo tworzy rozwiązanie przy użyciu metody greedySA. Następnie wykonuje się pętla tak długo, jak nie zostało przekroczone kryterium stopu lub jeśli temperatura nie przekroczyła minimalnej dopuszczalnej wartości. W pętli tej losowany jest indeks miasta oraz miejsce do jego umieszczenia, zamiana zostaje wykonana metodą pomocniczą insert. Losowanie to trwa tak długo, jak długo koszt nowej ścieżki jest większy od najlepszej wyznaczonej oraz gdy nieprzekroczona została ustalona ilość pętli. Po jej wykonaniu w przypadku wyznaczeniu lepszej ścieżki, to rozwiązanie staje się globalnym. W przeciwnym wypadku, jeśli warunek na prawdopodobieństwo jest spełniony, to wyznaczone lokalne rozwiązanie, mimo gorszego kosztu, staje się rozwiązaniem globalnym. Na koniec temperatura zostaje zaktualizowana poprzez przemnożenie jej przez ustalony współczynnik schładzania. Metoda ostatecznie wypisuje wyznaczoną ścieżkę wraz z czasem wyznaczenia oraz kosztem,
- Metoda pomocnicza greedySA odpowiedzialna jest za wyznaczenie ścieżki, przy użyciu prostego algorytmu zachłannego. Algorytm ten polega na szukaniu najkrótszych ścieżek wychodzących z rozpatrywanego miasta. Następnie brane pod uwagę są miasta do których prowadzą wyznaczone poprzednio ścieżki. Metoda modyfikuje wektor ścieżki przekazany w argumencie oraz zwraca koszt wyznaczonej ścieżki,

- Metoda pomocnicza insert umieszcza miasto o indeksie indexToDelete, w miejscu o indeksie indexToPut w ścieżce path.
- Metoda pomocnicza countPath oblicza i zwraca koszt podanej ścieżki.

3 Eksperymenty

3.1 Plan eksperymentu

Program wczytywać będzie pliki ftv47.atsp, ftv170.atsp, rgb403.atsp oraz będzie na pobranych danych wykonywać algorytmy Tabu Search oraz symulowanego wyżarzania. Każdy algorytm dla każdego pliku zostanie przeprowadzony 10 razy. Kryterium stopu zostanie ustawione dla każdego pliku odpowiednio na 2, 4 i 6 minut. Przedstawione zostaną najlepsze rozwiązania wraz z czasem wyznaczenia. Dodatkowo zostanie sporządzony wykres błędu funkcji czasu dla uśrednionych wyników dla każdego pliku. Błąd zostanie określony jako błąd względny, z racji posiadanych informacji o najlepszych rozwiązaniach dla każdego pliku. Czas mierzony jest z wykorzystaniem biblioteki std::chrono.

3.2 Wyniki

3.2.1 Tabu Search

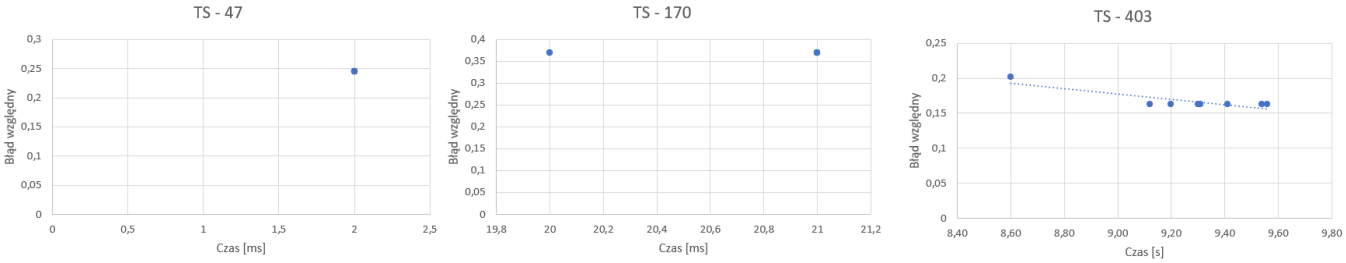
Rozmiar	Optymalnie	Znalezione	Czas	Błąd
47	1772	2207	2 ms	0,245485
170	2755	3771	20 ms	0,368784
403	2465	2866	5 s	0,162677

Tabela 1: Najlepsze wyniki algorytmu Tabu Search

Najlepsze ścieżki:

- ftv47.atsp: 0 -> 25 -> 1 -> 9 -> 33 -> 27 -> 28 -> 24 -> 4 -> 29 -> 30 -> 3 -> 6 -> 10 -> 11 -> 8 -> 32 -> 7 -> 31 -> 5 -> 18 -> 17 -> 35 -> 14 -> 16 -> 45 -> 20 -> 38 -> 37 -> 40 -> 47 -> 26 -> 2 -> 41 -> 43 -> 42 -> 22 -> 19 -> 44 -> 15 -> 23 -> 12 -> 34 -> 13 -> 46 -> 36 -> 39 -> 21 -> 0,
- ftv170.atsp: 0 -> 1 -> 2 -> 77 -> 73 -> 170 -> 49 -> 50 -> 51 -> 52 -> 53 -> 43 -> 55 -> 54 -> 58 -> 59 -> 60 -> 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 -> 102 -> 103 -> 117 -> 118 -> 119 -> 120 -> 121 -> 122 -> 123 -> 162 -> 101 -> 104 -> 114 -> 109 -> 113 -> 164 -> 127 -> 126 -> 125 -> 124 -> 129 -> 128 -> 130 -> 131 -> 132 -> 133 -> 134 -> 141 -> 6 -> 7 -> 8 -> 9 -> 10 -> 76 -> 74 -> 75 -> 11 -> 12 -> 18 -> 19 -> 20 -> 21 -> 22 -> 23 -> 26 -> 27 -> 28 -> 29 -> 30 -> 31 -> 33 -> 34 -> 156 -> 40 -> 39 -> 38 -> 37 -> 35 -> 36 -> 157 -> 41 -> 155 -> 42 -> 45 -> 44 -> 46 -> 47 -> 48 -> 168 -> 72 -> 78 -> 82 -> 79 -> 80 -> 81 -> 3 -> 4 -> 5 -> 169 -> 111 -> 110 -> 71 -> 158 -> 32 -> 16 -> 17 -> 24 -> 25 -> 150 -> 160 -> 151 -> 152 -> 142 -> 143 -> 144 -> 140 -> 149 -> 148 -> 147 -> 137 -> 136 -> 138 -> 135 -> 139 -> 14 -> 13 -> 15 -> 159 -> 161 -> 115 -> 116 -> 146 -> 145 -> 112 -> 0,

- rbg403.atsp: f0 -> 267 -> 46 -> 23 -> 14 -> 62 -> 13 -> 205 -> 204 -> 24 -> 36 -> 32 -> 274 -> 83 -> 33 -> 376 -> 68 -> 38 -> 270 -> 19 -> 57 -> 42 -> 287 -> 107 -> 61 -> 257 -> 43 -> 34 -> 295 -> 50 -> 56 -> 394 -> 225 -> 58 -> 8 -> 29 -> 64 -> 3 -> 2 -> 386 -> 47 -> 112 -> 322 -> 272 -> 41 -> 101 -> 55 -> 11 -> 59 -> 76 -> 310 -> 52 -> 119 -> 73 -> 217 -> 9 -> 84 -> 281 -> 77 -> 156 -> 147 -> 78 -> 226 -> 154 -> 79 -> 69 -> 245 -> 81 -> 22 -> 21 -> 82 -> 247 -> 249 -> 54 -> 85 -> 5 -> 172 -> 35 -> 67 -> 94 -> 86 -> 75 -> 10 -> 27 -> 116 -> 87 -> 26 -> 307 -> 143 -> 25 -> 365 -> 96 -> 114 -> 353 -> 263 -> 364 -> 303 -> 122 -> 40 -> 341 -> 102 -> 213 -> 278 -> 198 -> 104 -> 203 -> 349 -> 115 -> 189 -> 109 -> 275 -> 210 -> 92 -> 51 -> 259 -> 97 -> 387 -> 384 -> 383 -> 389 -> 187 -> 123 -> 192 -> 66 -> 74 -> 168 -> 126 -> 160 -> 15 -> 167 -> 130 -> 165 -> 359 -> 133 -> 269 -> 355 -> 6 -> 137 -> 256 -> 120 -> 392 -> 351 -> 293 -> 88 -> 65 -> 124 -> 339 -> 374 -> 28 -> 182 -> 333 -> 48 -> 328 -> 317 -> 155 -> 20 -> 264 -> 358 -> 327 -> 159 -> 323 -> 214 -> 401 -> 216 -> 200 -> 224 -> 166 -> 199 -> 177 -> 352 -> 170 -> 284 -> 290 -> 174 -> 185 -> 12 -> 176 -> 31 -> 305 -> 37 -> 212 -> 181 -> 346 -> 338 -> 184 -> 45 -> 363 -> 289 -> 188 -> 299 -> 49 -> 215 -> 309 -> 193 -> 250 -> 279 -> 196 -> 282 -> 350 -> 132 -> 243 -> 134 -> 288 -> 291 -> 315 -> 127 -> 191 -> 103 -> 175 -> 90 -> 72 -> 266 -> 180 -> 271 -> 357 -> 139 -> 30 -> 314 -> 218 -> 145 -> 372 -> 178 -> 325 -> 254 -> 223 -> 319 -> 125 -> 60 -> 44 -> 230 -> 382 -> 320 -> 332 -> 296 -> 146 -> 381 -> 236 -> 227 -> 144 -> 239 -> 70 -> 347 -> 39 -> 164 -> 342 -> 18 -> 246 -> 80 -> 301 -> 228 -> 297 -> 251 -> 113 -> 391 -> 258 -> 345 -> 173 -> 344 -> 336 -> 292 -> 329 -> 294 -> 100 -> 98 -> 194 -> 324 -> 306 -> 354 -> 16 -> 280 -> 242 -> 117 -> 148 -> 390 -> 171 -> 63 -> 337 -> 262 -> 276 -> 298 -> 362 -> 110 -> 326 -> 138 -> 377 -> 366 -> 321 -> 379 -> 283 -> 277 -> 398 -> 396 -> 370 -> 330 -> 331 -> 378 -> 209 -> 356 -> 334 -> 367 -> 368 -> 252 -> 371 -> 128 -> 304 -> 95 -> 211 -> 162 -> 308 -> 179 -> 237 -> 311 -> 312 -> 313 -> 7 -> 136 -> 316 -> 153 -> 318 -> 222 -> 231 -> 286 -> 273 -> 395 -> 265 -> 240 -> 219 -> 158 -> 152 -> 190 -> 142 -> 300 -> 232 -> 261 -> 248 -> 335 -> 131 -> 93 -> 169 -> 234 -> 207 -> 99 -> 244 -> 206 -> 149 -> 201 -> 89 -> 241 -> 195 -> 106 -> 340 -> 141 -> 183 -> 17 -> 129 -> 255 -> 151 -> 161 -> 157 -> 1 -> 360 -> 361 -> 135 -> 186 -> 53 -> 400 -> 285 -> 108 -> 111 -> 369 -> 163 -> 302 -> 118 -> 373 -> 4 -> 375 -> 399 -> 71 -> 208 -> 268 -> 150 -> 235 -> 121 -> 220 -> 202 -> 385 -> 221 -> 197 -> 388 -> 348 -> 343 -> 253 -> 140 -> 238 -> 229 -> 393 -> 105 -> 397 -> 260 -> 91 -> 233 -> 380 -> 402 -> 0



Rysunek 1: Wykresy rozkładu błędu od czasu rozstrzygnięcia wyniku dla wszystkich plików i algorytmu tabu search

Na wykresach często widnieją pojedyncze punkty, ponieważ zwracane rozwiązania były identyczne i czasami różniły się jedynie małymi wartościami czasu.

3.2.2 Symulowane wyżarzanie

Dla algorytmu symulowanego wyżarzania ustalono następujące współczynniki:

- Temperatura początkowa = 500,
- Minimalna temperatura = 0.00001,
- Współczynnik schładzania = 0.99,

Rozmiar	Optymalnie	Znalezione	Czas	Błąd
47	1772	1971	5 ms	0,112302
170	2755	3486	25 ms	0,265336
403	2465	2967	16 ms	0,203651

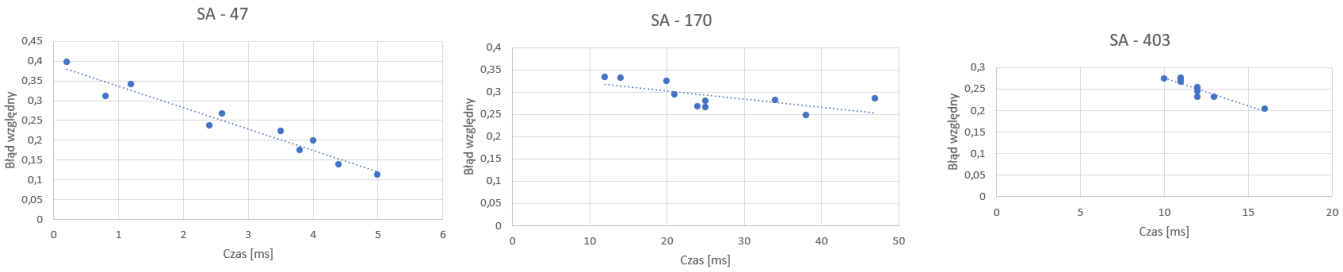
Tabela 2: Najlepsze wyniki algorytmu Symulowanego Wyżarzania

Ostatnia wyznaczona temperatura wynosiła zazwyczaj około 0.0000095.

Najlepsze ścieżki:

- ftv47.atsp: 0 -> 25 -> 1 -> 9 -> 33 -> 27 -> 2 -> 28 -> 24 -> 4 -> 29 -> 3 -> 8 -> 32 -> 31 -> 30 -> 5 -> 6 -> 10 -> 11 -> 18 -> 17 -> 12 -> 7 -> 23 -> 34 -> 13 -> 46 -> 36 -> 14 -> 35 -> 15 -> 16 -> 45 -> 20 -> 38 -> 37 -> 47 -> 26 -> 42 -> 41 -> 43 -> 22 -> 19 -> 44 -> 39 -> 21 -> 40 -> 0
- ftv170.atsp: 0 -> 1 -> 2 -> 77 -> 73 -> 170 -> 49 -> 51 -> 52 -> 53 -> 43 -> 55 -> 54 -> 58 -> 59 -> 60 -> 68 -> 67 -> 167 -> 70 -> 87 -> 85 -> 86 -> 83 -> 84 -> 69 -> 66 -> 63 -> 64 -> 56 -> 57 -> 62 -> 61 -> 65 -> 88 -> 153 -> 154 -> 89 -> 90 -> 91 -> 96 -> 97 -> 98 -> 95 -> 94 -> 92 -> 93 -> 166 -> 107 -> 106 -> 105 -> 165 -> 163 -> 99 -> 100 -> 102 -> 118 -> 119 -> 120 -> 121 -> 122 -> 123 -> 162 -> 101 -> 104 -> 115 -> 116 -> 117 -> 103 -> 114 -> 113 -> 164 -> 127 -> 126 -> 125 -> 124 -> 129 -> 128 -> 130 -> 131 -> 132 -> 133 -> 134 -> 6 -> 7 -> 8 -> 9 -> 10 -> 76 -> 74 -> 75 -> 11 -> 12 -> 18 -> 19 -> 20 -> 21 -> 29 -> 22 -> 23 -> 26 -> 27 -> 28 -> 30 -> 31 -> 33 -> 34 -> 156 -> 40 -> 39 -> 38 -> 35 -> 36 -> 157 -> 41 -> 155 -> 42 -> 45 -> 44 -> 46 -> 47 -> 48 -> 168 -> 72 -> 78 -> 82 -> 79 -> 80 -> 81 -> 3 -> 4 -> 5 -> 169 -> 110 -> 109 -> 108 -> 71 -> 50 -> 158 -> 32 -> 24 -> 25 -> 150 -> 161 -> 160 -> 151 -> 152 -> 142 -> 149 -> 148 -> 147 -> 137 -> 138 -> 139 -> 140 -> 141 -> 15 -> 159 -> 16 -> 17 -> 111 -> 112 -> 135 -> 136 -> 146 -> 145 -> 144 -> 143 -> 14 -> 13 -> 37 -> 0
- rbg403.atsp: 0 -> 267 -> 46 -> 23 -> 14 -> 62 -> 13 -> 205 -> 204 -> 24 -> 36 -> 32 -> 274 -> 83 -> 33 -> 376 -> 68 -> 38 -> 270 -> 19 -> 57 -> 42 -> 287 -> 107 -> 61 -> 257 -> 43 -> 34 -> 295 -> 50 -> 56 -> 312 -> 394 -> 225 -> 58 -> 8 -> 29 -> 64 -> 3 -> 2 -> 386 -> 47 -> 112 -> 322 -> 272 -> 41 -> 101 -> 55 -> 11 -> 28 -> 4 -> 140 -> 59 -> 153 -> 310 -> 52 -> 119 -> 73 -> 65 -> 9 -> 84 -> 281 -> 77 -> 156 -> 147 -> 78 -> 226 -> 154 -> 79 -> 69 -> 245 -> 81 -> 22 -> 21 -> 82 -> 247 -> 249 -> 85 -> 260 -> 35 -> 221 -> 67 -> 94 -> 314 -> 75 -> 27 -> 116 -> 87 -> 26 -> 313 -> 143 -> 114 -> 353 -> 263 -> 95 -> 303 -> 122 -> 40 -> 341 -> 374 -> 278 -> 198 -> 104 -> 349 -> 115 -> 189 -> 109 -> 275 -> 210 -> 92 -> 51 -> 259 -> 97 -> 384 -> 383 -> 93 -> 389 -> 187 -> 123 -> 192 -> 66 -> 74 -> 168 -> 335 -> 352 -> 157 -> 1 -> 333 -> 30 -> 131 -> 169 -> 165 -> 359 -> 206 -> 355 -> 6 -> 137 -> 54 -> 234 -> 120 -> 392 -> 351 -> 293 -> 88 -> 393 -> 124 -> 339 -> 304 -> 268 -> 261 -> 162 -> 48 -> 328 -> 317 -> 155 -> 20 -> 356 -> 358 -> 327 -> 159 -> 323 -> 214 -> 401 -> 216 -> 200 -> 224 -> 166 -> 199 -> 229 -> 86 -> 170 -> 284 -> 290 -> 174 -> 185 -> 388 ->

12 -> 176 -> 31 -> 305 -> 37 -> 132 -> 243 -> 178 -> 212 -> 181 -> 346 -> 338 -> 184 -> 45 -> 233 -> 380 -> 289 -> 188 -> 299 -> 49 -> 215 -> 309 -> 193 -> 250 -> 279 -> 196 -> 282 -> 350 -> 167 -> 105 -> 288 -> 291 -> 315 -> 127 -> 191 -> 139 -> 71 -> 209 -> 72 -> 266 -> 180 -> 271 -> 357 -> 235 -> 121 -> 113 -> 395 -> 218 -> 300 -> 171 -> 102 -> 273 -> 211 -> 98 -> 207 -> 254 -> 223 -> 319 -> 125 -> 60 -> 44 -> 230 -> 382 -> 320 -> 232 -> 265 -> 286 -> 296 -> 361 -> 146 -> 381 -> 236 -> 227 -> 144 -> 239 -> 70 -> 347 -> 39 -> 164 -> 25 -> 318 -> 342 -> 18 -> 246 -> 80 -> 238 -> 316 -> 228 -> 297 -> 251 -> 391 -> 258 -> 201 -> 118 -> 400 -> 397 -> 173 -> 344 -> 331 -> 292 -> 329 -> 294 -> 325 -> 264 -> 324 -> 306 -> 354 -> 16 -> 280 -> 248 -> 348 -> 213 -> 363 -> 362 -> 151 -> 148 -> 390 -> 175 -> 63 -> 337 -> 262 -> 89 -> 217 -> 134 -> 110 -> 326 -> 138 -> 377 -> 285 -> 321 -> 379 -> 202 -> 90 -> 398 -> 396 -> 370 -> 330 -> 343 -> 194 -> 301 -> 256 -> 108 -> 255 -> 152 -> 7 -> 399 -> 307 -> 378 -> 308 -> 5 -> 237 -> 311 -> 252 -> 371 -> 136 -> 222 -> 231 -> 160 -> 365 -> 240 -> 130 -> 158 -> 190 -> 100 -> 373 -> 142 -> 244 -> 117 -> 332 -> 336 -> 17 -> 141 -> 269 -> 126 -> 10 -> 99 -> 15 -> 149 -> 182 -> 241 -> 195 -> 106 -> 369 -> 103 -> 183 -> 129 -> 135 -> 219 -> 161 -> 360 -> 96 -> 186 -> 364 -> 367 -> 76 -> 111 -> 340 -> 133 -> 163 -> 368 -> 302 -> 372 -> 179 -> 334 -> 375 -> 53 -> 128 -> 208 -> 150 -> 203 -> 220 -> 197 -> 283 -> 277 -> 385 -> 345 -> 276 -> 298 -> 242 -> 387 -> 145 -> 253 -> 177 -> 172 -> 91 -> 402 -> 366 -> 0



Rysunek 2: Wykresy rozkładu błędu od czasu rozstrzygnięcia wyniku dla wszystkich plików i algorytmu symulowanego wyżarzania

4 Wnioski

Testy zwróciły oczekiwane rozwiązania, oba algorytmy zwracają optymalne rozwiązania. Implementacja algorytmu Tabu Search zdaje się często zwracać bardzo podobne wyniki po podobnym czasie. Natomiast implementacja algorytmu symulowanego wyżarzania zwraca zróżnicowane wyniki, które w przypadku coraz dłuższych wyliczeń, zwracają rozwiązania z coraz mniejszym błędem względnym. W miarę zwiększania się danych problemu tabu search zwracał optymalne rozwiązanie po coraz większym czasie. Symulowane wyżarzanie zazwyczaj oblicza wyniki o mniejszym błędzie względnym w krótszym czasie w porównaniu do przeszukiwania tabu.

Literatura

[1] https://en.wikipedia.org/wiki/Tabu_search

[2] https://www.ii.uni.wroc.pl/ prz/2011lato/ah/opracowania/szuk_tabu.opr.pdf

[3] http://155.158.112.25/ algorytmyewolucyjne/materialy/algorytm_symulowanego_wyzarzania.pdf

[4] http://iswiki.if.uj.edu.pl/images/2/20/AiSD_22._Symulowane_wyżarzanie_%28problem_komiwojażera%29.pdf

[5] Wykłady dr inż. Tomasza Kapłona z kursu Projektowania Efektywnych Algorytmów