ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: AUTOMATYKA z.117 Nr kol. 1337

Ewa SKUBALSKA-RAFAJŁOWICZ
Politechnika Wrocławska

SZYBKI ALGORYTM ROZWIĄZYWANIA DUŻYCH ZADAŃ KOMIWOJAŻERA

Streszczenie. W pracy sformułowano heurystyczny algorytm rozwiązywania zadania komiwojażera z odległościami euklidesowymi, który jest algorytmem lokalnych popraw i ma złożoność obliczeniową $O(n \log n)$, a przypadku implementacji na $O(n \cdot \log n^{-1/2})$ procesorach równoległych - $O(\log n^{3/2})$. Pozwala to efektywnie rozwiązywać duże zadania komiwojażera. W pracy przedstawiono wyniki eksperymentów numerycznych.

FAST ALGORITHM FOR LARGE TRAVELLING SALESMAN PROBLEMS

Summary. A heuristic for Travelling Salesman Problem with Euclidean distance will be given. The algorithm makes local improvements in starting tour and runs in $O(n \log n)$ time on serial processor or in $O(\log n^{3/2})$ on $O(n \cdot \log n^{-1/2})$ parallel processors. Thus the algorithm is able to solve efficiently large traveling salesman problems. Some numerical experiments are given.

1. Wstęp

Zagadnienie komiwojażera (TSP) jest klasycznym problemem w optymalizacji dyskretnej. Obszerną monografią na temat TSP jest książka [14]. Zadanie TSP jest problemem NP-zupełnym, stąd konieczność rozwijania algorytmów przybliżonych, działających w czasie wielomianowym. Zadanie TSP, w którym poszczególne punkty są określone poprzez ich współrzędne w przestrzeni d-wymiarowej, a odległości między nimi są wyznaczane zgodnie z metryką euklidesową, nazywane jest w skrócie ETSP.

Zadanie ETSP także, nawet w przypadku dwuwymiarowym, jest problemem NP-zupełnym. W związku z tym szczególnie intensywnie rozwijane są prace nad konstrukcjami algorytmów heurystycznych pozwalających w krótkim czasie wyznaczyć dobre, choć niekoniecznie optymalne rozwiązanie problemu. Znane są obecnie zastosowania TSP wymaga-

jące utworzenia trasy zawierającej ponad 1.2 miliona punktów. Szeroki przegląd szybkich algorytmów przybliżonego rozwiązywania zadania komiwojażera z odległościami euklidesowymi (i innymi typu geometrycznego) znajduje się w pracach [3], [19].

W niniejszej pracy przedstawiony zostanie szybki algorytm rozwiązywania zadań ETSP o złożoności obliczeniowej $O(n \log n)$ w wersji szeregowej, który w naturalny sposób może być realizowany w wersji równoległej. Eksperymenty obliczeniowe pokazują, że algorytm ten zapewnia uzyskanie podobnych rozwiązań jak jeden z najlepszych algorytmów aproksymacyjnych TSP zwany metodą włączania najdalszego wierzchołka [24].

2. Przegląd metod przybliżonego rozwiązywania zadań ETSP

Algorytmy przybliżone są jedynymi realizowalnymi w praktyce metodami rozwiązywania trudnych zadań kombinatorycznych o dużych rozmiarach. Opracowano i przebadano bardzo wiele algorytmów przybliżonych rozwiązywania TSP. Można je podzielić na dwie klasy [24], [20]. Pierwsza klasa obejmuje algorytmy tworzące dobre rozwiązanie przybliżone od początku, bez korzystania ze znajomości innego rozwiązania dopuszczalnego. Czasami tego typu algorytmy są nazywane algorytmami aproksymacyjnymi. Druga klasa to algorytmy modyfikujące (poprawiające) istniejące rozwiązanie. W praktyce na ogół łączy się ze sobą oba podejścia, używając algorytmu z klasy pierwszej do wygenerowania rozwiązania startowego dla algorytmu modyfikującego. Przegląd znanych algorytmów heurystycznych ETSP znajduje się w pracach [20], [16], [14], [13], [24].

Jednym z najbardziej znanych algorytów tworzących trasę TSP jest algorytm najbliższego sąsiada (NN), w którym trasa TSP jest konstruowana w następujący sposób: jako węzeł początkowy przyjmuje się dowolny punkt spośród tworzących zadanie punktów $x_1, x_2, \ldots, x_n$. Następnie znajduje się węzeł nie włączony do tej pory do trasy, który znajduje się najbliżej w stosunku do ostatnio przyłączonego do trasy wierzchołka. Po ulokowaniu w trasie wszystkich węzłów ostatni z nich łączony jest z punktem startowym trasy. Złożoność obliczeniowa algorytmu NN jest rzędu $O(n^2)$. W pracy [3] Bentley opisał wersję algorytmu NN, w którym konstruowanie trasy rozpoczyna się od kilku punktów startowych i tworzy się równocześnie kilka fragmentarycznych tras zgodnie z regułą NN. Następnie łączy się te fragmenty w jedną dużą trasę. Algorytm ten pozwala często uzyskiwać nieco lepsze wyniki niż oryginalny algorytm NN.

Do klasy algorytmów aproksymacyjnych ETSP typu NN można również zaliczyć algorytmy wykorzystujące krzywe wypełniające SFC [18], [21]. W algorytmach tych łączy się punkty zgodnie z ich kolejnością występowania na wybranej krzywej wypełniającej, a więc łączy się najbliższych sąsiadów zgodnie z odległością mierzoną wzdłuż krzywej. Złożoność obliczeniowa tego typu algorytmów jest rzędu $O(nlogn)$, podobnie jak złożoność obliczeniowa zadania ETSP, w którym wszystkie punkty leżą na jednej prostej. Naturalnie takie rozwiązanie jest optymalne tylko przy metryce związanej z SFC. W przypadku metryki euklidesowej i przy założeniu, że współrzędne punktów tworzących problem ETSP są realizacjami niezależnych zmiennych losowych o rozkładzie równomiernym w kwadracie $[0,1]\times[0,1]$, wartość oczekiwana długości trasy ETSP otrzymana za pomocą SFC (krzywa Sierpińskiego) jest rzędu $0.956\cdot\sqrt{n}$ [18]. Rozwiązanie ETSP otrzymane za pomocą SFC jest nie gorsze niż $O(logn)\cdot T^*$ [18], [22]. Idea liniowego porządkowania punktów z przestrzeni wielowymiarowej pojawia się również algorytmach w ETSP wykorzystujących sieci neuronowe typu Kohonena [4], [6], [7]. Rozwiązywanie za ich pomocą dużych zadań ETSP wymaga ogromnych nakładów obliczeniowych [7].

Jak do tej pory, za najbardziej efektywne uważane są klasyczne algorytmy aproksymacyjne takie jak metoda włączania najdalszego wierzchołka FI oraz metoda najtańszego włączania wierzchołka CI [20], [24]. W algorytmach typu włączania najpierw wybierany jest dowolny wierzchołek, który stanowi początek trasy. Następnie, zgodnie z ustaloną regułą, znajduje się kolejny nie występujący do tej pory w trasie węzeł. Po wybraniu wierzchołka szukane jest w trasie miejsce (para punktów), w którym włączenie wybranego wierzchołka spowoduje najmniejszy przyrost długości trasy TSP. Regułą wyboru może być: losowanie punktu (RI), wybór wierzchołka najbliższego w stosunku do punktów już umieszczonych w trasie (NI), wybór wierzchołka najbardziej oddalonego w stosunku do punktów już znajdujących się w trasie (FI), wybór wierzchołka, którego włączenie spowoduje najmniejszy przyrost długości trasy (CI). Złożoność obliczeniowa tego typu algorytów jest zależna od reguły wyboru wierzchołka. W przypadku algorytmu FI, który będzie dalej wykorzystywany w niniejszej pracy, jest ona rzędu $O(n^2)$. Eksperymenty obliczeniowe wskazują, że najlepszym prostym algorytmem aproksymacyjnym, ze względu na wartości średnich długości tras, jest właśnie algorytm FI [2], [24],[3].

Poza algorytmami włączania można również korzystać z prostszych schematów aproksymacyjnych zwanych metodami dołączania [20], [3], w których wybrany (zgodnie z jedną

z wymienionych wcześniej reguł) punkt dołączany jest jako bezpośredni sąsiad poprzednio wybranego wierzchołka. Uproszczenie to powoduje zmniejszenie nakładów obliczeniowych, niemniej jednak prowadzi na ogół do uzyskania dłuższych tras. Złożoność obliczeniową wymienionych procedur można poprawić wykorzystując specjalne struktury danych lub wprowadzając pewne uproszczenia do algorytmów. W pracy Bentleya [3] pokazano implementację algorytmu FI o złożoności obliczeniowej $O(n(logn)^2)$. Wersje algorytmów zaproponowane w pracy [3] wymagają stworzenia specjalnej struktury danych w postaci drzewa (k-dtree), co pociąga za sobą konieczność poniesienia nakładów obliczeniowych rzędu $O(n \log n)$, podobnego jak w przypadku algorytmu SFC. Koszt wyznaczenia najbliższego sąsiada przy przeszukiwaniu wielowymiarowego drzewa (k-dtree) jest w najgorszym przypadku rzędu $O(n)$, jednak można pokazać, że wartość oczekiwana liczby obliczeń odległości między punktami potrzebna do znalezienia najbliższego sąsiada jest $O(1)$. W konsekwencji pozwala to zrealizować algorytm NN w czasie (w sensie średnim) rzędu $O(n \log n)$.

Inna klasa algorytmów aproksymacyjnych oparta jest na budowie minimalnego drzewa rozpinającego [5], prowadzą one jednak do dłuższych tras ETSP niż najlepszy pod tym względem algorytm FI, który daje średnią długość trasy ETSP (naturalnie przy równomiernym rozkładzie punktów w kwadracie) rzędu $0.805\sqrt{n} + 0.23$, [3], [17]. Mówiąc o metodach rozwiązywania ETSP należy również wspomnieć algorytm Karpa [14] oraz jego uogólnienie na przypadek wielowymiarowy – algorytm Haltona i Terady [11]. Mają one bardzo dobre własności asymptotyczne, lecz otrzymanie za ich pomocą dobrych tras wymaga znacznych nakładów obliczeniowych.

Druga klasa algorytmów przybliżonych TSP obejmuje metody lokalnej optymalizacji, modyfikujące istniejące rozwiązanie. Jedną z najbardziej znanych metod z tej klasy jest algorytm k-wymian [15], [20],[24], [3],[14], w którym kolejne rozwiązanie jest otrzymywane poprzez usunięcie k wybranych połączeń między punktami w trasie i zastąpienie ich innymi k połączeniami (tak by zachować nadal zamknięty cykl puntów). Rozwiązanie, które nie może ulec poprawie w wyniku takiej procedury, nazywane jest k-optymalnym. Już w przypadku $k = 3$ algorytm ma złożoność obliczeniową rzędu $O(n^3)$. W pracach [23], [3] rozważane są różne modyfikacje algorytmów typu k wymian o mniejszej złożoności obliczeniowej.

W przypadku stosowania metod lokalnej optymalizacji pojawia się problem wycho-

dzenia rozwiązania z minimum lokalnego. Próby zbliżenia się do minimum globalnego wiążą się z rozwijaniem algorytmów typu tabu [8], algorytmów genetycznych TSP [9], [12], algorytmów symulowanego wyżarzania [10] oraz zastosowaniem związanych z nim maszyn Boltzmanna [1], [14]. Wspomniane podejścia na razie nie nadają się do szybkiego rozwiązywania dużych zadań ETSP (zbyt duży nakład obliczeniowy), choć takie próby są podejmowane [7], [4].

3. Opis algorytmu

Przedstawiony w pracy algorytm jest algorytmem typu lokalnych popraw, który w zamierzeniu ma generować trasy o podobnej długości jak algorytm aproksymacyjny FI (włączania najdalszego wierzchołka), przy złożoności obliczeniowej co najwyżej rzędu $O(n\log n)$. Jako punkt wyjścia – rozwiązanie początkowe – przyjęto uporządkowanie punktów zgodnie z ich występowaniem na SFC typu Sierpińskiego [21],[18]. Przy założeniu że współrzędne punktów tworzących problem ETSP są realizacjami niezależnych zmiennych losowych o rozkładzie równomiernym w kwadracie $[0,1] \times [0,1]$, wartość oczekiwana długości trasy ETSP przy użyciu algorytmu SFC jest rzędu $0.956 \cdot \sqrt{n}$ [18]. Eksperymenty obliczeniowe przeprowadzone przez autorkę wskazują, że średnia długość trasy komiwojażera (przy danych pochodzących z trójwymiarowej kostki jednostkowej) jest rzędu $1.08 \cdot n^{2/3}$. Algorytm SFC można w naturalny sposób zrównoleglić, gdyż pozycja dowolnego punktu na krzywej nie zależy w żaden sposób od położenia pozostałych punktów i może być wyznaczana oddzielnie.

Główna część przedstawionego w pracy algorytmu polega na lokalnym poprawianiu fragmentów istniejącej trasy. Za pomocą okienka o długości k (liczba punktów) pobierany jest z aktualnej trasy fragment obejmujący k kolejnych punktów. Punkt pierwszy i ostatni z wybranej części trasy zachowują swoje pozycje, natomiast pozostałe $k-2$ punkty są szeregowane zgodnie z opisanym uprzednio, klasycznym algorytmem FI. Jeżeli długość pobranego fragmentu trasy uległa skróceniu, to stare uporządkowanie fragmentu trasy zostaje zastąpione nowym, w przeciwnym przypadku fragment trasy pozostaje bez zmian. W kolejnym kroku algorytmu okienko pobierające zostaje przesunięte o p pozycji (w prawo lub w lewo zależnie od przyjętej konwencji), przy czym $1 \leq p \leq k$. Ponownie reguła FI zostaje użyta do próby skonstruowania lepszego porządku w trasie obejmującej

pobrane punkty. Budowa nowego uporządkowania wymaga wykonania za każdym razem $O(k^2)$ operacji. Procedura powtarzana jest aż do momentu, gdy okienko zostanie przesunięte wzdłuż całej trasy ETSP. Przesuwanie okienka z krokiem mniejszym niż k pozwala szybko wymieniać punkty pomiędzy poszczególnymi fragmentami trasy. Przyspiesza to uzyskiwanie dobrych rozwiązań. Przesunięcie okienka równe jego długości ($p = k$) powoduje, że wymiana punktów pomiędzy podzbiorami związanymi z poszczególnymi fragmentami tras może nastąpić dopiero, gdy zmienimy punkt startowy okienka pobierającego. Takie rozwiązanie jest konieczne w przypadku implementacji równoległej algorytmu.

Minimum lokalne osiągane jest, gdy przesunięcie okienka wzdłuż pełnej trasy i zastosowanie metody FI nie doprowadzi do uzyskania poprawy. Dalsze kontynuowanie procedury na pewno nie przyniesie skrócenia długości trasy komiwojażera. Natomiast zmiana początkowej pozycji okienka o l punktów w prawo lub w lewo, przy $l \neq p$, może przyczynić się do wyjścia z lokalnego minimum i uzyskania lepszej trasy ETSP.

Przesunięcie okienka wzdłuż całej trasy wymaga wykonania łącznie $\lceil n/p \rceil \cdot O(k^2)$ operacji. W realizowanej wersji algorytmu przyjęto $p = \lceil k/2 \rceil$, co pozwala przyspieszyć ewentualną wymianę punktów pomiędzy kolejnymi fragmentami trasy. Ponadto założono, że zmiana pozycji startowej okienka będzie następowała zaraz po wykonaniu jednego pełnego obiegu okienka wzdłuż trasy, bez czekania na osiągnięcie minimum lokalnego. Wprawdzie kontynuowanie procedury bez zmiany punktu startowego okienka może przynieść jeszcze pewną poprawę, lecz eksperymenty obliczeniowe pokazały, iż wcześniejsza zmiana punktu przyjmowanego umownie jako początek trasy prowadzi w konsekwencji do uzyskiwania lepszych rozwiązań. Możliwe, że osiągnięcie optimum lokalnego utrudnia ponowne wyjście z niego poprzez tak drobną modyfikację, jaką jest zmiana punktu startowego okienka pobierającego. Sprawdzenie wszystkich $n - \lceil n/p \rceil$ możliwych położeń powodowałaby zbyt gwałtowny wzrost potrzebnych nakładów obliczeniowych, stąd ograniczono się do przesuwania pozycji początkowej okienka o $l = 1, 2, \ldots, p-1$ pozycji. W konsekwencji cała procedura, która dalej nazywana będzie algorytmem LFI, wymaga wykonania $\lceil k/2 \rceil \cdot \lceil 2n/k \rceil \cdot O(k^2) = O(nk^2)$ operacji obliczeniowych. Jest oczywiste, że k powinno rosnąć wraz ze wzostem n. Przy założeniu że k zmienia się zgodnie z regułą $k_n = \alpha \cdot \log n^{1/2}$, gdzie α jest pewną stałą, całkowita złożoność obliczeniowa algorytmu LFI jest rzędu $O(nlogn)$. Tak opisana procedura LFI musi być wykonana kilkakrotnie, zanim osiągnięte zostanie związane z nią minimum lokalne (podobnie jak w algorytmie

k-opt [15]). Eksperymenty obliczeniowe pokazały, iż przeciętnie dwu-, trzykrotne powtórzenie algorytmu LFI prowadzi do uzyskania lokalnego optimum.

Algorytm LFI

1. Wyznacz początkowe uszeregowanie punktów $(X_1, X_2, \ldots, X_n)$ zgodnie z krzywą wypełniającą SFC [18], [21].
2. Ustal wartości parametrów przesuwu okienka k i p, $1 \leq p \leq k$.
3. Ustal punkt startowy okienka pobierającego fragment trasy, przesuwając cyklicznie wszystkie punkty w trasie o l pozycji w lewo $l = 0, 1, 2, \cdots, p-1$.
4. Wykonaj kolejne kroki (a),(b),(c),(d),(e) dla $i = 0, 1, 2, \cdots, \lceil n/p \rceil - 1$.
5. Pobierz z trasy $(X_{(1)}, X_{(2)}, \cdots, X_{(n)}) = (X_{1+l}, X_{2+l}, \cdots, X_{n+l})$.
   fragment $((X_{(ip+1)}, X_{(ip+2)}, \cdots, X_{(ip+k)})$
6. Wyznacz jego długość
$$dl = \sum_{j=1}^{k-1} ||X_{(ip+j)} - X_{(ip+j+1)}||$$
7. Uporządkuj punkty $X_{(ip+2)}, X_{(ip+3)}, \cdots, X_{(ip+k-1)}$ zgodnie z regułą FI, przyjmując punkt $X_{(ip+1)}$ za punkt początkowy, a punkt $X_{(ip+k)}$ za punkt końcowy.
8. Wyznacz długość nowego uporządkowania $((X'_{(ip+1)}, X'_{(ip+2)}, \ldots, X'_{(ip+k)})$
$$dln = \sum_{j=1}^{k-1} ||X'_{(ip+j)} - X'_{(ip+j+1)}||$$
9. Jeżeli $dln < dl$, zmień pobrany fragment trasy na nowo wyznaczone uporządkowanie.

Zamiast reguły FI można zastosować lokalnie inny algorytm heurystyczny ETSP. W pracy zbadano także metodę L2W, w której zamiast algorytmu FI zastosowano algorytm 2-optymalnych popraw.

Ponieważ rozwiązaniem startowym algorytmu LFI jest uporządkowanie zgodne z krzywą SFC, to współczynnik najgorszego przypadku jest tutaj również rzędu $O(logn)$. Zaletą przedstawionego algorytmu LFI jest również możliwość równoległego przetwarzania wszystkich rozłącznych fragmentów trasy. Najprostszym rozwiązaniem jest podzielenie trasy na n/k lub $n/(k-1)$ fragmentów zawierających po k punktów. Każdy z tych fragmentów może być poprawiany niezależnie od pozostałych. Wymianę punktów pomiędzy fragmentami trasy zapewnia powtórzenie całej procedury przy innym punkcie startowym podziału trasy. Przy przyjęciu $k = O(\log n^{1/2})$ i założeniu takiej samej jak w

przypadku implementacji szeregowej liczby powtórzeń prowadzi to do konieczności użycia $O(n \cdot \log n^{-1/2})$ procesorów w czasie rzędu $O(\log n^{3/2})$.

4. Wyniki eksperymentów numerycznych

W części tej przedstawiono wyniki obliczeń numerycznych. W tablicy 1 umieszczono długości drogi ETSP otrzymane za pomocą badanych algorytmów heurystycznych: SFC,LFI,L2W oraz ich połączeń LFI-L2W i L2W-LFI, w których algorytm LFI zastosowano do trasy uzyskanej uprzednio za pomocą algorytmu L2D (L2W-LFI), lub rozwiązanie wynikłe z zastosowania metody LFI poprawiono następnie za pomocą algorytmu L2W (LFI-L2W). Badania przeprowadzono dla liczby punktów (miast) równej $n = 128, 512, 1024, 10000$. Długość pobieranych fragmentów trasy ustalono eksperymentalnie, rozwiązując wielokrotnie relatywnie małe zadania ETSP ($n = 128$). Ustalono, że wartość $k = 16$ dla algorytmu LFI oraz $k = 18$ dla algorytmu L2W, gwarantuje otrzymanie rozwiązań porównywalnych ze średnimi wynikami uzyskiwanymi przy użyciu klasycznych metod FI oraz 2-optymalnych. Położenie punktów ETSP generowano losowo z rozkładem równomiernym w kwadracie ($d = 2$) lub kostce 3-wymiarowej ($d = 3$). Odległości między punktami obliczane były zgodnie z normą euklidesową. Wszystkie eksperymenty polegały na na 10-krotnym losowaniu zestawów n punktów, każdorazowym obliczeniu długości drogi ETSP wyznaczonej przez badane heurezy i uśrednieniu wyników. W tablicy 1 przedstawiono wyniki dla $d = 2, 3$ i różnych rozmiarów zadań.

Poza średnią długością drogi wyznaczano również wartości współczynników $T_{sr}/n^{(d-1)/d}$, gdzie T_{sr} oznacza średnią długość trasy ETSP uzyskiwaną przez poszczególne metody. Wyniki te zamieszczono w tablicy 2.

Stosując metodę najmniejszych kwadratów do oszacowania odpowiednich współczynników otrzymano, że średnia długość drogi ETSP ($d = 2$) uzyskana metodą LFI jest równa $0.813 \cdot \sqrt{n}$ lub $0.796 \cdot \sqrt{n} + 1.19$. Porównanie otrzymanych długości dróg ze średnią długością trasy ETSP uzyskiwaną za pomocą algorytmu FI (naturalnie przy równomiernym rozkładzie punktów w kwadracie) – rzędu $0.805\sqrt{n} + 0.23$ [3] – prowadzi do wstępnego wniosku, że algorytm LFI może z powodzeniem zastąpić algorytm LI. Zastosowanie dodatkowo algorytmu L2W w niewielkim stopniu poprawia otrzymywane rozwiązanie.

Eksperymenty obliczeniowe wykonywane były na komputerze Pentium 100Mhz. Pro-

Tablica 1

Długość drogi ETSP uzyskiwana za pomocą heurez LFI,L2W,L2W-LFI oraz LFI-L2W

d	N	SFC	LFI	L2W	L2W-LFI	LFI-L2W
2	128	10.896	9.40	9.81		9.37
3	128	27.28	21.10	21.74	21.13	21.09
2	512	21.72	18.59	19.34	18.46	18.56
3	512	70.58	54.70	56.47	55.14	54.53
2	1024	30.63	26.17	27.58	26.17	26.12
2	10000	95.37	80.61	85.96	80.54	80.60

Tablica 2

Relatywna długość drogi ETSP uzyskiwanej za pomocą heurez LFI,L2W,L2W-LFI oraz LFI-L2W względem $n^{(d-1)/d}$

d	N	SFC	LFI	L2W	L2W-LFI	LFI-L2W
2	128	0.963	0.831	0.867		0.828
3	128	1.074	0.831	0.856	0.832	0.830
2	512	0.960	0.822	0.855	0.816	0.820
3	512	1.103	0.855	0.882	0.862	0.852
2	1024	0.957	0.818	0.862	0.818	0.818
2	10000	0.954	0.806	0.860	0.805	0.803

gramy realizujące badane algorytmy napisane zostały w języku Mathematica. Na podstawie eksperymentów obliczeniowych czas obliczeń algorytmu LFI został aproksymowany jako $0.11 \cdot n \cdot \log(n)$ dla $128 \leq n \leq 10000$. Na podstawie czasu obliczania odległości można stwierdzić, że, w porównaniu z komputerem VAX-8550, używany komputer był około 55 razy wolniejszy. Rozwiązanie problemu ETSP za pomocą algorytmu LFI z $n = 10000$ pochłonęło około 3.5 godziny. Dla komputera VAX-8550 byłby to co najwyżej czas rzędu 4 minut. Czas ten można by wydatnie skrócić, pisząc implementację algorytmu LFI w języku C. Dla porównania czasy obliczeń zadania ETSP z $n = 1000$ za pomocą sieci neuronowej i algorytmu symulowanego wyżarzania na komputerze Vax3600 były odpowiednio

rzędu 40 i 60 minut [7], a otrzymane długości drogi ETSP były nieco większe niż wartość oczekiwana dla algorytmu FI. Podstawową zaletą niniejszego algorytmu jest jednak możliwość zrównoleglania obliczeń. Algorytm niewątpliwie wymaga dalszych badań zarówno teoretycznych, jak eksperymentów obliczeniowych, jednak wyniki osiągnięte dla zadań o dużych rozmiarach ($n = 10000$) są obiecujące.

LITERATURA

1. Aarts E.H.L.. and Korst J.H.M.: Boltzmann machines for travelling salesman problem. EJOR, Vol.39, 1989, pp.79-95.
2. Adrabiński A. and Sysło M.M.: Computational experiments with some approximation algorithms for the traveling salesman problem. Zastosowania Matematyki, Vol.18, 1983, pp.91–95.
3. Bentley J.J.: Fast algorithms for geometric traveling salesman problems. ORSA Journal on Computing, Vol.4, 1992, pp.387–411.
4. Budinich M.: A self-organizing neural network for the traveling salesman problem that is competitive with simmulated annealing. Neural Computation, Vol.8, 1996, pp.416–424.
5. Christofides N. and Eilon S.: Algorithms for large-scale travelling salesman problems. Oper. Res. Quart., Vol.23, 1972, pp.511–518.
6. Durbin R. and Willshow D.: An analogue approach to the travelling salesman problem using an elastic net method. Nature (London), Vol.336, 1987, pp.689–691.
7. Favata F. and Walker R.: A study of the applications of Kohonen-type neural networks to travelling salesman problem. Biol. Cybern., Vol.64, 1991, pp.463–468.
8. Glover F.: Tabu search: a tutorial. Interfaces, Vol.20, 1990, pp.74–94.
9. Goldberg D.E.: Algorytmy genetyczne i ich zastosowania, WNT, Warszawa 1995.
10. Golden B.L. and Skiscim C.C.: Using simulated annealing to solve routing and location problems. Naval Research Logistics Quorterly, Vol.33, 1986, pp.261–279.
11. Halton J.H. and Terada R.: A fast algorithm for the Euclidean traveling salesman problem, optimal with probability one. SIAM J. on Comput., Vol.11, 1982, pp.28–46.
12. Homaifar A., Guan S., and Liepins G.E.: Schema analysis of the traveling salesman problem using genetic algorithms. Complex Systems, Vol.6, 1992, pp.533–552.
13. Laporte G.: The traveling salesman problem: an overviev of exact and approximate algorithms. EJOR, Vol.59, 1992, pp.231–247.

14. Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G., and Shmoys D. B.: The Traveling Salesman Problem. Wiley, New York, 1985.

15. Lin S. and Kerninghan B.: An effective heuristic algorithm for the traveling salesman problem. Operations Research, Vol.21, 1973, pp.498–516.

16. Norback J.P. and Love R.F.: Heuristic for the Hamiltonian path problem in Eucldean two space. J. of the Operational Res. Soc., Vol.30, 1979, pp.363–368.

17. Ong H.L. and Huang H.C.: Asymptotic expected performance of some tsp heuristics: An empirical evaluation. EJOR, Vol.43, 1989, pp.231–238.

18. Platzman L.K. and Bartholdi J.J. III.: Spacefilling curves and the planar travelling salesman problem. Journal of the ACM, Vol.36, 1989, pp.719–737.

19. Reinelt G.: Fast heuristics for large geometric traveling salesman problems. ORSA J. on Comput., Vol.4, 1992, pp.206–217.

20. Rosenkrantz D.J., Stearns R.E., and Lewis P.M II.: An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput., Vol.6, 1977, pp.563–581.

21. Skubalska-Rafajlowicz E.: Algorytm rozwiązywania zadania komiwojażera z metryką euklidesową. Zesz. Nauk.Pl.Śl., Ser. Automatyka, 114, 1994, pp.241–250.

22. Steele J.M.: Efficacy of spacefilling heuristics in euclidean combinatorial optimization. Operations Reserch Letters, Vol.8, 1989, pp.237–239.

23. Stewart W.R. Jr.: Accelerated branch exchange heuristics for symmetric traveling salesman problems. Networks, Vol.17, 1987, pp.423–437.

24. Sysło M.M., Deo N., and Kowalik J.: Algorytmy dyskretnej optymalizacji. PWN, Warszawa 1993.

Recenzent: Prof. dr hab. inż. Jerzy Klamka

Wpłynęło do Redakcji do 30.06.1996 r.

Abstract

A fast heuristic for Travelling Salesman Problem with Euclidean distance is proposed. An algorithm is described, which makes local improvements in starting tour given by $O(n \log n)$ space-filling curve heuristic. The starting route is divided on almost disjoint paths. Each path is to be improved by the known farthest insertion algorithm. The algorithm runs in $O(n \log n)$ time on serial processor or in $O(\log n^{3/2})$ on $O(n \cdot \log n^{-1/2})$ parallel processors.

The algorithm is able to solve efficiently large Travelling Salesman Problems. Numerical experiments with 128, 512, 1024 and 10000 of i.d.d. points in two and three dimensions are given.