STUDIA INFORMATICA 2014
Volume 35 Number 1 (115)

Jacek WIDUCH, Artur KLYTA
Politechnika Śląska, Instytut Informatyki

ALGORYTM HYBRYDOWY WIELOKROTNEGO STARTU DLA ROZWIĄZYWANIA PROBLEMU SEKWENCYJNEGO UPORZĄDKOWANIA

Streszczenie. Problem sekwencyjnego uporządkowania (SOP) jest podobny do asymetrycznego problemu komiwojażera. Celem jest wyznaczenie w skierowanym grafie ważonym ścieżki Hamiltona o minimalnej wadze, przy dodatkowym spełnieniu relacji pierwszeństwa wierzchołków. W niniejszej pracy zaprezentowano algorytm hybrydowy wielokrotnego startu rozwiązywania problemu SOP. Algorytm ten jest połączeniem algorytmów symulowanego wyżarzania i lokalnej optymalizacji. Dodatkowo przedstawiono wyniki przeprowadzonych badań eksperymentalnych.

Słowa kluczowe: problem sekwencyjnego uporządkowania, problem NP–trudny, cykl Hamiltona, ścieżka Hamiltona, algorytm heurystyczny, symulowane wyżarzanie, lokalna optymalizacja, algorytm hybrydowy

A MULTISTART HYBRIDALGORITHM TO SOLVING THE SEQUENTIAL ORDERING PROBLEM

Summary. The sequential ordering problem (SOP) is similar to the asymmetric traveling salesman problem. The goal is to find a minimum weight Hamiltonian path on a directed weighted graph satisfying precedence relationships among the vertices. In the paper, a multistart hybrid algorithm to solving SOP is presented. The algorithm based on simulated annealing algorithm and local optimization method. Apart from that results of experimental tests are presented.

Keywords: sequential ordering problem, NP–hard problem, Hamiltonian cycle, Hamiltonian path, heuristic algorithm, simulated annealing, local optimization, hybrid algorithm

1. Wstęp

Jednym z problemów teorii grafów jest problem komiwojażera (TSP, ang. *traveling salesman problem*) [8, 9, 16, 21, 25, 26, 27, 28, 29, 39, 40, 44]. W problemie TSP danych jest: n miast oraz odległości między każdą parą miast, przy czym odległości są symetryczne. Celem jest wyznaczenie marszruty komiwojażera, który każde z miast musi odwiedzić dokładnie jeden raz i powrócić do miasta, z którego wyjechał. Poszukiwana jest marszruta o minimalnej długości. Problem TSP jest rozwiązywany z użyciem algorytmów grafowych. Sieć miast opisana jest za pomocą pełnego nieskierowanego grafu ważonego, zawierającego n wierzchołków reprezentujących miasta, krawędzie natomiast reprezentują odległości między miastami. W grafie należy wyznaczyć cykl Hamiltona o minimalnej wadze.

Problem TSP jest określany mianem problemu symetrycznego. Istnieje również wersja asymetryczna tego problemu (ATSP, ang. *asymmetric traveling salesman problem*), w której odległość miasta i od miasta j może być różna od odległości miasta j od miasta i. Problem ATSP w teorii grafów modelowany jest więc za pomocą skierowanego grafu ważonego.

Wersja ATSP poddawana jest czasem modyfikacji, zgodnie z którą dla pewnych par miast zdefiniowana jest relacja pierwszeństwa. Niech będzie dana relacja pierwszeństwa $i \rightarrow j$ oznaczająca, że miasto i należy odwiedzić przed odwiedzeniem miasta j. Dla tak zdefiniowanej relacji miasto i nazywane jest miastem poprzednikiem, a miasto j nazywane jest miastem następnikiem. Jedną z interpretacji relacji pierwszeństwa może być sytuacja, w której towar dostarczany do miasta j komiwojażer otrzymuje w mieście i.

W problemie sekwencyjnego uporządkowania (SOP, ang. *sequential ordering problem*) sformułowanym w 1988 roku danych jest n miast oraz dane są odległości między miastami, przy czym odległości nie są symetryczne [10]. Zdefiniowane są także relacje pierwszeństwa dla pewnych par miast oraz dane są miasta początkowe i końcowe. Zadanie polega na wyznaczeniu marszruty z miasta początkowego do miasta końcowego, minimalizując jej długość. Każde z miast należy odwiedzić dokładnie jeden raz oraz powinny być spełnione relacje pierwszeństwa.

Problem TSP należy do grupy problemów NP–zupełnych, a problem ATSP do grupy problemów NP–trudnych. Problem SOP poprzez redukcję do problemu ATSP jest również problemem NP–trudnym. Dowody przynależności wymienionych problemów do poszczególnych klas złożoności dostępne są w pracach [5, 8, 21, 36, 37].

Powstało wiele prac przedstawiających rozwiązanie problemu SOP. Jest on rozwiązywany z użyciem różnego rodzaju technik, do których można zaliczyć: algorytmy mrowiskowe [14, 15, 37], algorytmy genetyczne [45], algorytmy podziału i ograniczeń [18, 35], algorytmy k-optymalne [13], algorytmy hybrydowe [5]. Do rozwiązywania problemu SOP stosowana jest także relaksacja Lagrange'a [4, 11, 12] czy też algorytmy równoległe [17, 44]. W litera-

turze rozważany jest także stochastyczny problem SOP [3]. W niniejszej pracy zaproponowano hybrydowy algorytm wielokrotnego startu rozwiązywania problemu SOP. Zaproponowany algorytm jest połączeniem algorytmu symulowanego wyżarzania i lokalnej optymalizacji.

Struktura niniejszej pracy przedstawia się następująco. W rozdziale 2 omówiony został problem SOP, przedstawiono jego definicję w odniesieniu do teorii grafów oraz zamieszczono analizę problemu. Rozdział 3 zawiera opis zaproponowanego algorytmu hybrydowego rozwiązywania problemu SOP. W rozdziale 4 zamieszczono wyniki przeprowadzonych badań eksperymentalnych z użyciem algorytmu omówionego w rozdziale 3. Podsumowanie pracy zamieszczono w rozdziale 5.

2. Problem sekwencyjnego uporządkowania

W teorii grafów problem SOP zdefiniowany jest w następujący sposób [13, 18]. Dany jest skierowany graf ważony $G = (V, E, c)$, gdzie $V = \{v_1, \ldots, v_n\}$ jest zbiorem zawierającym n wierzchołków, $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$ jest zbiorem łuków, natomiast $c : E \to N$ jest funkcją wagową. Dane są także wierzchołki początkowy v_s i końcowy v_e. Dodatkowo dany jest prosty acykliczny skierowany graf $G_\mathrm{p} = (V, E_\mathrm{p})$, opisujący relacje pierwszeństwa, gdzie $E_\mathrm{p} \subset E$. Łuk $(v_i, v_j) \in E_\mathrm{p}$ opisuje relację pierwszeństwa $v_i \to v_j$. Należy wyznaczyć ścieżkę Hamiltona z wierzchołka v_s do wierzchołka v_e, minimalizując wagę ścieżki i uwzględniając relacje pierwszeństwa. Można spotkać się z pracami, w których problem SOP definiowany jest jako problem ATSP z relacjami pierwszeństwa [5, 18, 36].

Tabela 1
Macierz wag opisująca graf G oraz macierz opisująca relacje pierwszeństwa reprezentowane grafem G_p

G	1	2	3	4	5
1	0	4	3	5	2
2	3	0	5	4	6
3	3	3	0	4	5
4	4	4	2	0	5
5	2	1	3	3	0

G_p	1	2	3	4	5
1	–	1	1	1	1
2	0	–	0	0	1
3	0	1	–	0	1
4	0	1	0	–	1
5	0	0	0	0	–

Niech przykładowo będą dane macierze opisujące grafy G i G_p (tabela 1), wierzchołkiem początkowym jest $v_s = 1$, a wierzchołkiem końcowym $v_e = 5$. Wartość 1 w komórce o współrzędnych $[j, k]$ w macierzy opisującej graf G_p oznacza istnienie relacji pierwszeństwa $j \to k$, a wartość 0 brak takiej relacji. Warto podkreślić, że graf G_p definiuje w sposób jednoznaczny wierzchołki początkowy i końcowy. Zgodnie z grafem G_p wierzchołek 1 należy odwiedzić

przed odwiedzeniem każdego z pozostałych wierzchołków grafu, czyli musi on być wierzchołkiem początkowym. Podobnie w przypadku wierzchołka 5, który należy odwiedzić po wcześniejszym odwiedzeniu każdego z pozostałych wierzchołków grafu G, tak więc jest on wierzchołkiem końcowym.

W grafie G (tabela 1) istnieją dwie ścieżki Hamiltona R_1 i R_2 z wierzchołka $v_s = 1$ do wierzchołka $v_e = 5$ o minimalnej wadze równej 15:

$$R_1 = \langle 1, \quad 2, \quad 4, \quad 3, \quad 5 \rangle \tag{1}$$

$$R_2 = \langle 1, \quad 3, \quad 2, \quad 4, \quad 5 \rangle \tag{2}$$

Ścieżki te nie są poprawnymi rozwiązaniami, ponieważ nie są spełnione dla nich relacje pierwszeństwa, zgodnie z którymi przed odwiedzeniem miasta 2 należy odwiedzić miasta 3 i 4 (tabela 1). Stąd rozwiązaniem problemu jest ścieżka R_3 o wadze 16:

$$R_3 = \langle 1, \quad 4, \quad 3, \quad 2, \quad 5 \rangle \tag{3}$$

W praktyce grafy G i G_p scalane są w jeden graf opisany za pomocą jednej macierzy [18]. Niech przykładowo graf G_p zawiera łuk (v_i, v_j). W takim przypadku usuwany jest z grafu G łuk o przeciwnym zwrocie, tj. (v_j, v_i). Łuk (v_j, v_i) nie może należeć do ścieżki będącej rozwiązaniem, gdyż nie byłaby spełniona relacja pierwszeństwa. W macierzy wag opisującej graf G zamiast wagi łuku (v_j, v_i) wpisywana jest wartość –1. W identyczny sposób postępuje się ze wszystkimi łukami grafu G_p. W tabeli 2 przedstawiono scalone grafy G i G_p opisane macierzami przedstawionymi w tabeli 1.

Tabela 2

Macierz wag opisująca scalone grafy G i G_p

G	1	2	3	4	5
1	0	4	3	5	2
2	–1	0	–1	–1	6
3	–1	3	0	4	5
4	–1	4	2	0	5
5	–1	–1	–1	–1	0

3. Opis algorytmu hybrydowego wielokrotnego startu rozwiązywania problemu SOP

Algorytm jest połączeniem algorytmów symulowanego wyżarzania i lokalnej optymalizacji. Algorytm symulowanego wyżarzania, zainspirowany metodą Monte Carlo [31], jest wariantem przeszukiwania lokalnego i został opublikowany w 1953 roku [30]. W pracach [7, 23] zaproponowano jego użycie do rozwiązywania problemów optymalizacyjnych. Działanie algorytmu symulowanego wyżarzania jest podobne do procesu wyżarzania stosowanego

w metalurgii. Jest to jeden z podstawowych procesów obróbki cieplnej stopów żelaza, który polega na ogrzewaniu wyżarzanego materiału do pewnej temperatury, a następnie powolnym jego schładzaniu. W wyniku tego procesu dochodzi do modyfikacji struktury krystalicznej wyżarzanego materiału, a celem tych zmian jest doprowadzenie materiału w stanie końcowym do stanu równowagi termodynamicznej. W algorytmie w analogiczny sposób dochodzi do generowania sekwencji kolejnych stanów materiału. Niech T będzie aktualną temperaturą, a i aktualnym stanem o energii E_i. Kolejny stan i+1 o energii E_{i+1} jest generowany na podstawie stanu i poprzez proste losowe przekształcenie elementarne i znajduje się on w sąsiedztwie bieżącego stanu i. O przejściu do kolejnego stanu decyduje różnica energii między stanami, tj. $\Delta E = E_{i+1} - E_i$. Jeżeli $\Delta E \leq 0$, to następuje przejście do staniu i+1, który staje się stanem aktualnym. W przypadku kiedy $\Delta E > 0$, przejście do stanu i+1 dokonywane jest z prawdopodobieństwem P określonym rozkładem Boltzmanna (4), gdzie k jest stałą Boltzmanna [1, 52]. Przedstawiony sposób przejścia pomiędzy stanami nazywany jest kryterium Metropolisa.

$$P = \exp\left(-\frac{\Delta E}{kT}\right) \tag{4}$$

Generowanie kolejnych stanów jest powtarzane wielokrotnie dla zadanej temperatury T, po czym dokonywana jest redukcja temperatury. W przypadku osiągnięcia $T = 0$ następuje przejście do stanu zamrożenia, w którym zaprzestaje się generowania kolejnych stanów. Dokonując podsumowania, algorytm symulowanego wyżarzania sprowadza się więc do losowego przeszukiwania sąsiedztwa aktualnego stanu. Dokładny opis idei działania algorytmu dostępny jest np. w pracach [32, 33, 41, 42].

Dokonując rozwiązania problemu z użyciem algorytmu symulowanego wyżarzania, konieczne jest zdefiniowanie:

- reprezentacji rozwiązania,
- sposobu generowania rozwiązania początkowego,
- celu i funkcji oceny rozwiązania,
- sąsiedztwa bieżącego rozwiązania,
- parametrów sterujących działaniem algorytmu.

3.1.1. Reprezentacja rozwiązania

Dla badanego problemu, w którym danych jest n miast o numerach 1, …, n, użyto reprezentacji ścieżkowej kodowania permutacyjnego [34]. Rozwiązanie reprezentowane jest za pomocą wektora n-elementowego $R[1, \ldots, n]$. Element $R[i]$ ($i = 1, \ldots, n$) zawiera numer miasta znajdującego się na pozycji i–tej w wyznaczanej drodze, przy czym na pozycji pierwszej i ostatniej znajduje się odpowiednio miasto początkowe i końcowe, tj. $R[1] = v_s$, $R[n] = v_e$. Na pozycjach 2, …, n–1 znajduje się więc permutacja miast ze zbioru $V \setminus \{v_s, v_e\}$. Należy zwró-

cić uwagę na to, że nie każda permutacja miast jest rozwiązaniem problemu, gdyż musi ona spełniać relacje pierwszeństwa. Dla tak określonej metody reprezentacji rozwiązania przestrzeń rozwiązań w przypadku pesymistycznym zawiera $(n–2)!$ rozwiązań.

3.1.2. Generowanie rozwiązania początkowego

W algorytmie w procesie generowania rozwiązania początkowego $R[1, \ldots, n]$ został zastosowany czynnik losowy. Ponieważ $R[1] = v_s$ i $R[n] = v_e$, zatem zachodzi konieczność wygenerowania losowej permutacji miast ze zbioru $V \setminus \{v_s, v_e\}$ umieszczonych na pozycjach $2, \ldots, n–1$ w rozwiązaniu R. Podczas generowania losowej permutacji miast należy uwzględnić relacje pierwszeństwa. W tym celu przy generowaniu miasta znajdującego się na pozycji i–tej tworzony jest multizbiór kandydatów miast V_i ($i = 2, \ldots, n–1$) dopuszczalnych do umieszczenia na tej pozycji. Tworzenie multizbioru V_i odbywa się w następujących krokach:

1. W multizbiorze V_i umieszczane są wszystkie miasta ze zbioru V, z wyjątkiem miasta końcowego v_e oraz miast znajdujących się na pozycjach $1, \ldots, i–1$ w rozwiązaniu R.
2. Dla relacji pierwszeństwa $v_j \rightarrow v_k$ ($v_k \neq v_e$), jeżeli miasto v_j nie znajduje się w rozwiązaniu R na żadnej z pozycji $1, \ldots, i–1$, to jest ono dodawane do multizbioru V_i.
3. Dla relacji pierwszeństwa $v_j \rightarrow v_k$ ($v_k \neq v_e$) z multizbioru V_i usuwane jest miasto v_k, jeżeli miasto v_j nie znajduje się w rozwiązaniu R na żadnej z pozycji $1, \ldots, i–1$.

Usunięcie z multizbioru V_i miasta v_k w kroku 3 gwarantuje poprawność wygenerowanego rozwiązania początkowego. Dla relacji pierwszeństwa $v_j \rightarrow v_k$, po usunięciu miasta v_k, nie ma możliwości wygenerowania błędnego rozwiązania, w którym miasto to zostanie odwiedzone przed miastem v_j. Ponadto dodanie do multizbioru V_i w kroku 2 dodatkowego miasta v_j zwiększa prawdopodobieństwo wylosowania tego miasta i wstawienia go na pozycję i–tą w rozwiązaniu R.

Tabela 3

Macierz relacji pierwszeństwa

	1	2	3	4	5	6	7	8	9
1	–	1	1	1	1	1	1	1	1
2	0	–	0	0	0	0	0	0	1
3	0	0	–	0	0	0	0	0	1
4	0	0	0	–	0	0	0	0	1
5	0	1	0	0	–	0	0	0	1
6	0	0	0	0	0	–	0	0	1
7	0	0	1	0	1	0	–	1	1
8	0	1	0	0	0	0	0	–	1
9	0	0	0	0	0	0	0	0	–

Niech przykładowo $v_s = 1$ i $v_e = 9$, a relacje pierwszeństwa opisane są macierzą (tabela 3). Rozpatrzone zostaną dwa przypadki losowania miasta na pozycję i–tą w rozwiązaniu R. Dokonując losowania miasta na pozycję $i = 2$ (rys. 1a), multizbiór ma postać: $V_i = \{4, 6, 7, 7, 7,$

7}. Do multizbioru nie należą miasta 2, 3, 5 i 8, ponieważ występują one w relacjach pierwszeństwa jako miasta następniki, a miasta poprzedniki nie zostały jeszcze odwiedzone w rozwiązaniu *R*. Miasto 7 występuje jako poprzednik w trzech relacjach pierwszeństwa, dlatego multizbiór zawiera 3 dodatkowe kopie tego miasta. Prawdopodobieństwo wylosowania miasta 7 na pozycję $i = 2$ wynosi 2/3, a miast 4 i 6 wynosi 1/6.

Rys. 1. Fragment generowanego rozwiązania początkowego *R*
Fig. 1. A part of generated initial solution *R*

Drugim przypadkiem jest sytuacja losowania miasta na pozycję $i = 4$ (rys. 1b), w którym multizbiór V_i zawiera miasta: $V_i = \{3, 4, 5, 5, 8, 8\}$. Do multizbioru nie należy miasto 2, ponieważ występuje ono w relacjach pierwszeństwa, zgodnie z którymi najpierw należy odwiedzić miasta 5 i 8, które nie należą jeszcze do rozwiązania. Z tego powodu multizbiór zawiera dwie kopie miast 5 i 8.

3.1.3. Cel i funkcja oceny rozwiązania

Celem jest minimalizacja drogi przebytej z v_s do v_e, z uwzględnieniem relacji pierwszeństwa, co można opisać zależnością (5). Zatem funkcja oceny $f = dist(v_s, v_e)$ rozwiązania przyporządkowuje drodze jej długość i na jej podstawie dokonywane jest porównywanie rozwiązań w algorytmie. Funkcja ta określa jakość rozwiązania.

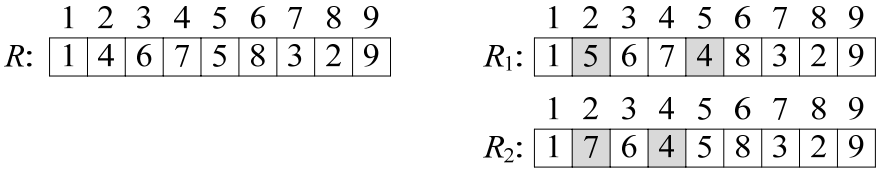
$$\min_R \{dist(v_s, v_e)\} \tag{5}$$

3.1.4. Sąsiedztwo bieżącego rozwiązania

Sąsiedztwo bieżącego rozwiązania *R* tworzą wszystkie rozwiązania powstałe przez zamianę pary miast znajdujących się na pozycjach 2, …, *n*–1 przy uwzględnieniu relacji pierwszeństwa. Maksymalnie sąsiedztwo zawiera N_n rozwiązań określonych zależnością (6).

$$N_n = \frac{(n-2)(n-3)}{2} \tag{6}$$

Niech przykładowo będzie dane rozwiązanie *R* (rys. 2) i macierz relacji pierwszeństwa (tabela 3). Zamiana miast na pozycjach 2 i 5 w rozwiązaniu *R* (miasta o numerach 4 i 5) powoduje utworzenie błędnego rozwiązania R_1, ponieważ miasto 7 nie zostanie odwiedzone przed odwiedzeniem miasta 5. Z kolei zamiana miast na pozycjach 2 i 4 w rozwiązaniu *R* (miasta o numerach 4 i 7) powoduje uzyskanie poprawnego rozwiązania R_2. W rozwiązaniu tym spełnione są wszystkie relacje pierwszeństwa.



Rys. 2. Aktualne rozwiązanie *R* i rozwiązania z jego sąsiedztwa
Fig. 2. A current *R* solution and solutions in its neighborhood

3.1.5. Parametry sterujące działaniem algorytmu

Do parametrów sterujących algorytmem należą:

- długość epoki,
- temperatura początkowa,
- metoda redukcji temperatury,
- warunek zakończenia.

Pierwszym z parametrów jest długość epoki L_i, czyli liczba sprawdzanych rozwiązań dla danej temperatury T_i. Jednym z rozwiązań jest przyjęcie długości epoki równej rozmiarowi sąsiedztwa [2], co zastosowano w niniejszym algorytmie. W tym przypadku górnym ograniczeniem rozmiaru epoki L_i jest zależność (6), czyli jest to wielkość równa $O(n^2)$.

$$T_0 = -\frac{\Delta f}{\ln(P_0)} \tag{7}$$

Temperatura początkowa T_0 wyznaczana jest w następujący sposób na podstawie początkowej wartości prawdopodobieństwa P_0 akceptacji rozwiązania gorszego [22] i zależności (7). Po wygenerowaniu rozwiązania początkowego R_0 następuje sprawdzenie L_i rozwiązań w jego sąsiedztwie. Podczas analizy rozwiązań wyznaczana jest wartość Δf będącą wartością średniego pogorszenia jakości zaakceptowanych rozwiązań.

Obliczanie temperatury początkowej można opisać następującym pseudokodem:

```
   wejście: R – rozwiązanie początkowe
            G – graf odległości między miastami i relacji pierwszeństwa
            P0 – początkowe prawdopodobieństwo akceptacji gorszego rozwiązania
   wyjście: T0 – wyznaczona temperatura początkowa
   procedure ComputeT0(R, G, P0);
 1: begin;
 2:   Δf := 0;
 3:   for i := 1 to Li do
 4:     Ri := i-te rozwiązanie z sąsiedztwa rozwiązania R;
 5:     if f(Ri) > f(R) then      // Ri jest gorszym rozwiązaniem
 6:       Δf := Δf + f(Ri) - f(R);
 7:     end if
 8:   end for
 9:   Δf := Δf / Li;
10:   T0 := temperatura początkowa obliczona według zależności (7);
11:   return T0;
12: end;
```

W wierszach 3–8 analizowanych jest L_i rozwiązań z sąsiedztwa rozwiązania R. Jeżeli rozwiązanie R_i z sąsiedztwa jest gorsze od aktualnego rozwiązania R, to następuje dodanie do Δf różnicy pomiędzy długością drogi w rozwiązaniach R_i i R (wiersz 6). Po przeanalizowaniu wszystkich rozwiązań obliczane jest średnie pogorszenie jakości rozwiązania (wiersz 9) oraz temperatura początkowa (wiersz 10), która zwracana jest jako wynik wykonania procedury (wiersz 11).

Na podstawie pracy [41] przyjęto redukcję temperatury zgodnie z zależnością (8), gdzie współczynnik redukcji temperatury α jest parametrem algorytmu. Jako warunek zakończenia przyjęto wykonanie m iteracji.

$$T := \alpha T \tag{8}$$

3.1.6. Pseudokod algorytmu

Opracowany algorytm hybrydowy wielokrotnego startu rozwiązywania problemu sekwencyjnego porządkowania został przedstawiony w postaci procedury *MS_H_SOP* opisanej następującym pseudokodem:

```
   wejście: k  - liczba uruchomień algorytmu
            G  - graf odległości między miastami i relacji pierwszeństwa
            P0 - początkowe prawdopodobieństwo akceptacji gorszego rozwiązania
            Plo - prawdopodobieństwo lokalnej optymalizacji rozwiązania R
            α  - współczynnik redukcji temperatury
   wyjście: R  - wyznaczona najkrótsza droga uwzględniająca relacje
                 pierwszeństwa
 1: procedure MS_H_SOP(k, G, P0, Plo, α)
 2: begin
 3:   R := ∅;
 4:   for i := 1 to k do
 5:     Ri := H_SOP(G, P0, Plo, α);
 6:     if f(Ri) < f(R) then      // znaleziono lepsze rozwiązanie
 7:       R := Ri;
 8:     end if
 9:   end for
10:   return R;
11: end;
```

W procedurze pamiętane jest najlepsze z dotychczas znalezionych rozwiązań R w poszczególnych uruchomieniach algorytmu. Wyznaczanie rozwiązań odbywa się w iteracji **for** (wiersze 4–9) przez k-krotne uruchomienie algorytmu hybrydowego opisanego procedurą *H_SOP* (wiersz 5). Po wyznaczeniu rozwiązania w procedurze *H_SOP* następuje sprawdzenie, czy wyznaczone przez nią rozwiązanie R_i jest lepsze niż dotychczas znalezione rozwiązanie R (wiersz 6), a jeżeli tak, to jest ono zapamiętywane (wiersz 7). Jako wynik wykonania procedury zwracane jest rozwiązanie R (wiersz 10) o minimalnej długości trasy, które zostało wyznaczone podczas k uruchomień algorytmu.

Procedurę *H_SOP* implementującą pojedyncze wykonanie algorytmu hybrydowego przedstawia pseudokod:

```
    wejście: G  - graf odległości między miastami i relacji pierwszeństwa
             P0 - początkowe prawdopodobieństwo akceptacji gorszego rozwiązania
             Plo - prawdopodobieństwo lokalnej optymalizacji rozwiązania R
             α  - współczynnik redukcji temperatury
    wyjście: R  - wyznaczona najkrótsza droga uwzględniająca relacje
                  pierwszeństwa
 1: procedure H_SOP(G, P0, Plo, α)
 2: begin
 3:   Rbest := ∅;        // najlepsze z dotychczas znalezionych rozwiązań
 4:   R := wyznacz rozwiązanie początkowe;
 5:   T := ComputeT0(R, G, P0);          // wyznaczenie temperatury początkowej
 6:   while not spełniony warunek zakończenia do
 7:     for i := 1 to Li do
 8:       Ri := rozwiązanie z sąsiedztwa bieżącego rozwiązania R;
 9:       if f(Ri) ≤ f(R) then    // rozwiązanie lepsze od aktualnego
10:         R := Ri;
11:       else if spełnione kryterium akceptacji rozwiązania Ri then
12:         if f(R) < f(Rbest) then
13:           Rbest := R;
14:         end if
15:         R := Ri;
16:         if spełnione kryterium lokalnej optymalizacji rozwiązania R then
17:           R := LocalOpt(R, G);
18:         end if
19:       end if
20:     end for
21:     T := αT;        // koniec epoki, redukcja temperatury
22:   end while
23:   if f(R) > f(Rbest) then
24:     R := Rbest;
25    end if
26:   return R;
27: end;
```

Najlepsze spośród wszystkich wyznaczonych rozwiązań w pojedynczym uruchomieniu algorytmu jest zapamiętywane w R_{best}. W części inicjalizacyjnej, zgodnie z zasadami opisanymi w punkcie 3.1.2, następuje wyznaczenie rozwiązania początkowego (wiersz 4) oraz temperatury początkowej (wiersz 5). Wyznaczanie rozwiązań odbywa się w iteracji **while** (wiersze 6–22) wykonywanej do momentu spełnienia warunku zakończenia omówionego w punkcie 3.1.5. W pojedynczej iteracji następuje sprawdzenie L_i rozwiązań wybieranych w sposób losowy z sąsiedztwa bieżącego rozwiązania (iteracja **for** w wierszach 7–20), po czym dokonywana jest redukcja temperatury (wiersz 21).

Po wyborze rozwiązania R_i z sąsiedztwa bieżącego rozwiązania R (wiersz 8) następuje porównanie długości tras obydwu rozwiązań. Jeżeli rozwiązanie R_i nie jest gorsze od rozwiązania R, to jest ono akceptowane jako aktualne rozwiązanie (wiersz 10). W przeciwnym przypadku jest ono akceptowane (wiersz 15) po spełnieniu kryterium akceptacji, zgodnie z prawdopodobieństwem opisanym zależnością (4). Ponieważ dochodzi do akceptacji rozwiązania gorszego, więc aby uniknąć utraty lepszego rozwiązania R, jest ono zapamiętywane jako rozwiązanie R_{best} (wiersz 13), jeżeli jest ono najlepsze z dotychczas znalezionych rozwiązań. Dodatkowo, w przypadku zaakceptowania rozwiązania gorszego dokonywana jest z prawdopodobieństwem P_{lo} próba lokalnej optymalizacji z użyciem procedury *LocalOpt* (wiersz 17). Po wyznaczeniu rozwiązań w iteracji **while** (wiersze 6–22) następuje porówna-

nie aktualnego rozwiązania R z rozwiązaniem R_{best} (wiersz 23) i jako wynik wykonania procedury *H_SOP* zwracane jest lepsze z nich (wiersz 26).

W procedurze *LocalOpt* dokonywana jest próba lokalnej optymalizacji rozwiązania R. Metoda ta polega na częściowym przeszukaniu sąsiedztwa bieżącego rozwiązania z użyciem algorytmu k–optymalnego [6, 19, 20, 27, 28, 43, 46], przy czym przyjęto $k = 3$. Zgodnie z tą metodą dokonywane jest sprawdzenie wszystkich możliwych tras powstałych przez zamianę trzech miast w bieżącym rozwiązaniu. Przy dokonywaniu zamiany rozpatrywane są wyłącznie poprawne rozwiązania spełniające relacje pierwszeństwa. Spośród tak wyznaczonych rozwiązań wybierane jest najlepsze rozwiązanie i jeżeli jest ono lepsze od rozwiązania bieżącego, to zastępuje ono bieżące rozwiązanie.

4. Wyniki badań eksperymentalnych

Badania eksperymentalne algorytmu opisanego w rozdziale 3 przeprowadzono z użyciem danych testowych zawartych w bibliotece TSPLIB [47]. Dla każdego zbioru wejściowego udostępniono najlepsze znane rozwiązanie, co pozwala ocenić badany algorytm. Badania eksperymentalne zostały przeprowadzone z użyciem komputera PC z procesorem AMD Athlon™ 64 Processor 3500+ (2.21 GHz) 1 GB RAM.

Badania składały się z dwóch części. Celem pierwszej części badań było określenie zależności czasu obliczeń i jakości wyznaczonego rozwiązania od wartości następujących parametrów algorytmu: współczynnika redukcji temperatury α, początkowej wartości prawdopodobieństwa P_0 akceptacji gorszego rozwiązania, liczby wykonanych iteracji m oraz wartości prawdopodobieństwa P_{lo} lokalnej optymalizacji. Badania przeprowadzono dla zestawów danych: ESC25, p43.1, prob.42 i ESC12 oraz wartości parametrów algorytmu:

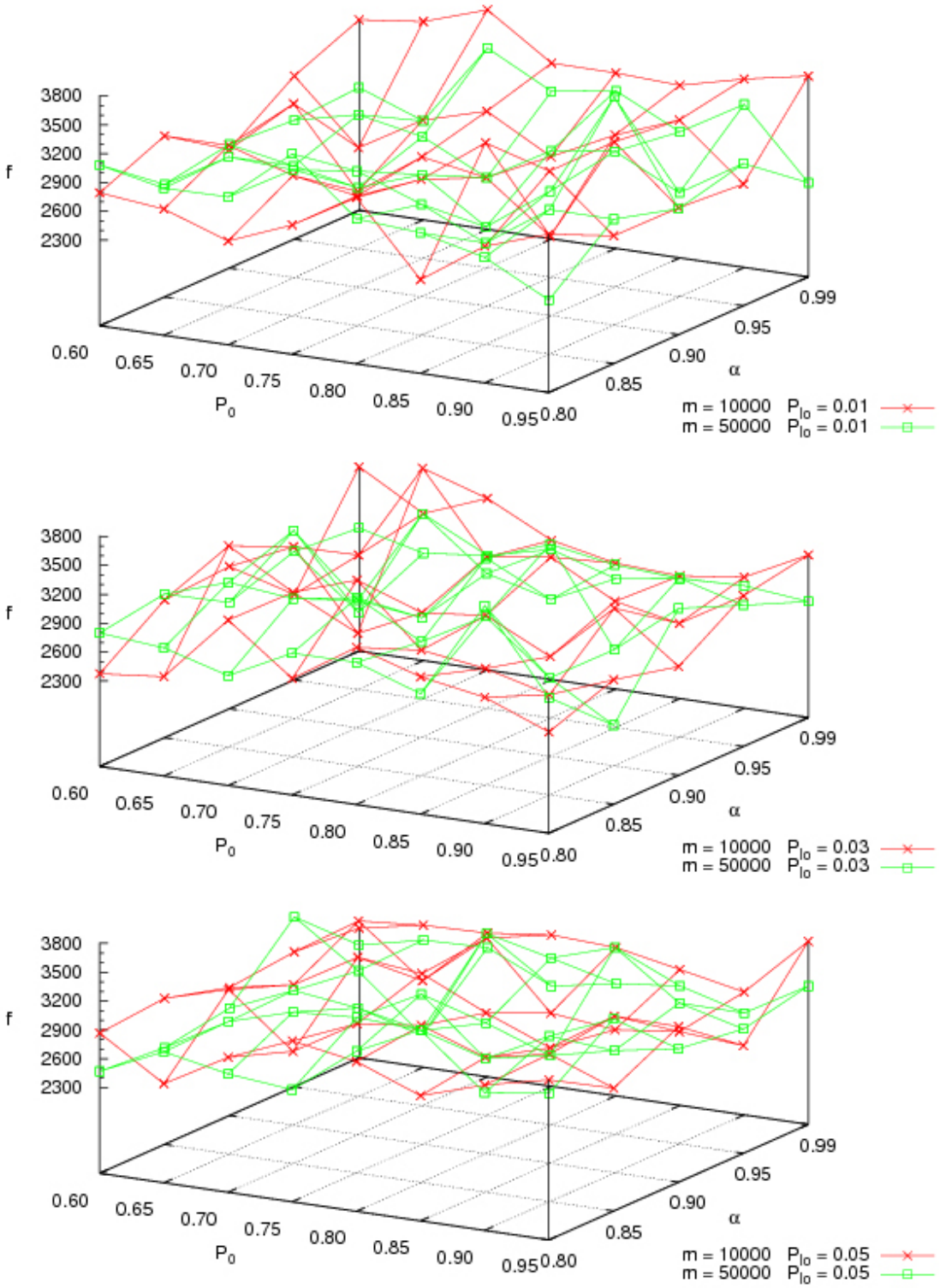
- współczynnik redukcji temperatury: $\alpha = 0.80; 0.85; 0.90; 0.95; 0.99$,
- początkowa wartość prawdopodobieństwa akceptacji gorszego rozwiązania: $P_0 = 0.60; 0.65; 0.70; 0.75; 0.80; 0.85; 0.90; 0.95$,
- liczba wykonanych iteracji: $m = 10000; 50000$,
- prawdopodobieństwo lokalnej optymalizacji: $P_{lo} = 0.01; 0.03; 0.05$.

Dla jednego zestawu danych przeprowadzono 240 testów dla wszystkich kombinacji wartości wymienionych parametrów. Dla ustalonej liczby iteracji m i prawdopodobieństwa lokalnej optymalizacji P_{lo} na wykresach 3D przedstawiono wpływ współczynnika redukcji temperatury α oraz początkowej wartości prawdopodobieństwa akceptacji gorszego rozwiązania P_0 na jakość wyznaczonego rozwiązania (rys. 3, 5 i 7) i na czas jego wyznaczania (rys. 4, 6, 8). Czas wyznaczania rozwiązania przedstawiono w formacie HH:MM.SS.

Na rys. 3 i 4 przedstawiono wyniki przeprowadzonych badań dla zestawu danych ESC25. Najlepszym uzyskanym rozwiązaniem było rozwiązanie wyznaczone w czasie 0:08.29 o jakości równej $f = 2140$ dla wartości parametrów: $\alpha = 0.95$, $P_0 = 0.70$, $m = 50000$, $P_{lo} = 0.05$. Uzyskane rozwiązanie jest o ok. 27% gorsze od optymalnego rozwiązania o jakości $f_{opt} = 1681$ [47]. Minimalny czas wyznaczania rozwiązania był równy 0:00.22 i uzyskano go podczas wyznaczania rozwiązania o jakości równej $f = 2846$ dla parametrów $\alpha = 0.95$, $P_0 = 0.70$, $m = 10000$, $P_{lo} = 0.01$. Maksymalny czas był równy 0:11.18 dla parametrów $\alpha = 0.85$, $P_0 = 0.75$, $m = 50000$, $P_{lo} = 0.05$ i wyznaczonego rozwiązania o jakości $f = 3028$. Najgorsze rozwiązanie o jakości równej $f = 6469$ zostało uzyskane dla parametrów $\alpha = 0.85$, $P_0 = 0.85$, $m = 10000$, $P_{lo} = 0.03$, a czas jego wyznaczania był równy 0:00.33.

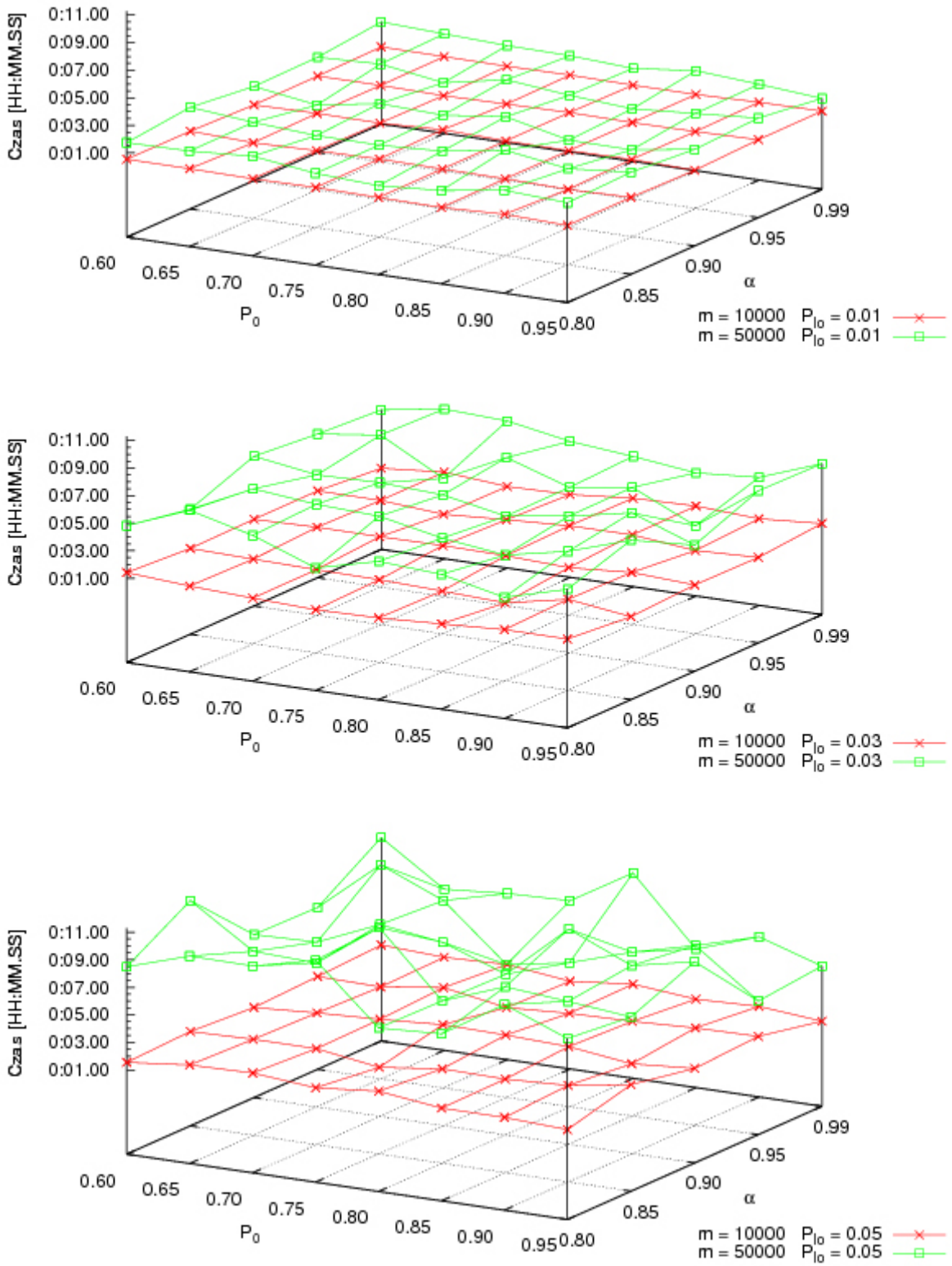
Wyniki przeprowadzonych badań dla zestawu danych p43.1 przedstawione zostały na rys. 5 i 6. Zgodnie z danymi przedstawionymi w [47], optymalnym rozwiązaniem jest rozwiązanie o jakości równej $f_{opt} = 27990$. Podczas przeprowadzonych badań najlepsze rozwiązanie uzyskane zostało dla parametrów $\alpha = 0.80$, $P_0 = 0.60$, $m = 50000$, $P_{lo} = 0.03$, a było nim rozwiązanie o jakości $f = 28225$ wyznaczone w czasie równym 1:10.06. Jest ono gorsze o prawie 1% od rozwiązania optymalnego. Najgorsze rozwiązanie zostało wyznaczone w czasie 0:10.11 dla parametrów algorytmu równych odpowiednio $\alpha = 0.99$, $P_0 = 0.95$, $m = 10000$, $P_{lo} = 0.03$, a jego jakość była równa $f = 29270$. Warto jednak zwrócić uwagę na to, że jest ono jedynie o ok. 4.5% gorsze od rozwiązania optymalnego. Minimalny czas wyznaczania rozwiązania był równy 0:03.45 dla wartości parametrów: $\alpha = 0.80$, $P_0 = 0.65$, $m = 10000$, $P_{lo} = 0.01$ i uzyskano rozwiązanie o jakości równej $f = 28730$. Z kolei maksymalny czas wyznaczania rozwiązania był równy 2:24.13 dla wartości parametrów $\alpha = 0.95$, $P_0 = 0.65$, $m = 50000$, $P_{lo} = 0.05$ i uzyskano rozwiązanie o jakości $f = 28400$.

Na wykresach na rys. 7 i 8 zostały przedstawione uzyskane wyniki dla zestawu danych prob.42. Optymalnym rozwiązaniem jest rozwiązanie o jakości równej $f_{opt} = 240$ [47]. Najlepsze rozwiązanie o jakości $f = 372$ uzyskano dla parametrów równych odpowiednio: $\alpha = 0.80$, $P_0 = 0.80$, $m = 10000$, $P_{lo} = 0.03$, a czas jego wyznaczania był równy 0:15.02. Jakość uzyskanego rozwiązania jest o ok. 55% gorsza od jakości rozwiązania optymalnego. Najgorszym rozwiązaniem było rozwiązanie o jakości równej $f = 636$, które zostało uzyskane dla parametrów: $\alpha = 0.99$, $P_0 = 0.85$, $m = 10000$, $P_{lo} = 0.01$, a czas jego wyznaczania był równy 0:06.17. Dla wartości parametrów: $\alpha = 0.90$, $P_0 = 0.95$, $m = 10000$, $P_{lo} = 0.01$ uzyskano minimalny czas wyznaczania rozwiązania, który był równy 0:02.42, oraz rozwiązanie o jakości $f = 511$. Maksymalny czas wyznaczania rozwiązania był równy 1:47.18 dla parametrów $\alpha = 0.80$, $P_0 = 0.60$, $m = 50000$, $P_{lo} = 0.05$, a uzyskanym rozwiązaniem było rozwiązanie o jakości $f = 460$.



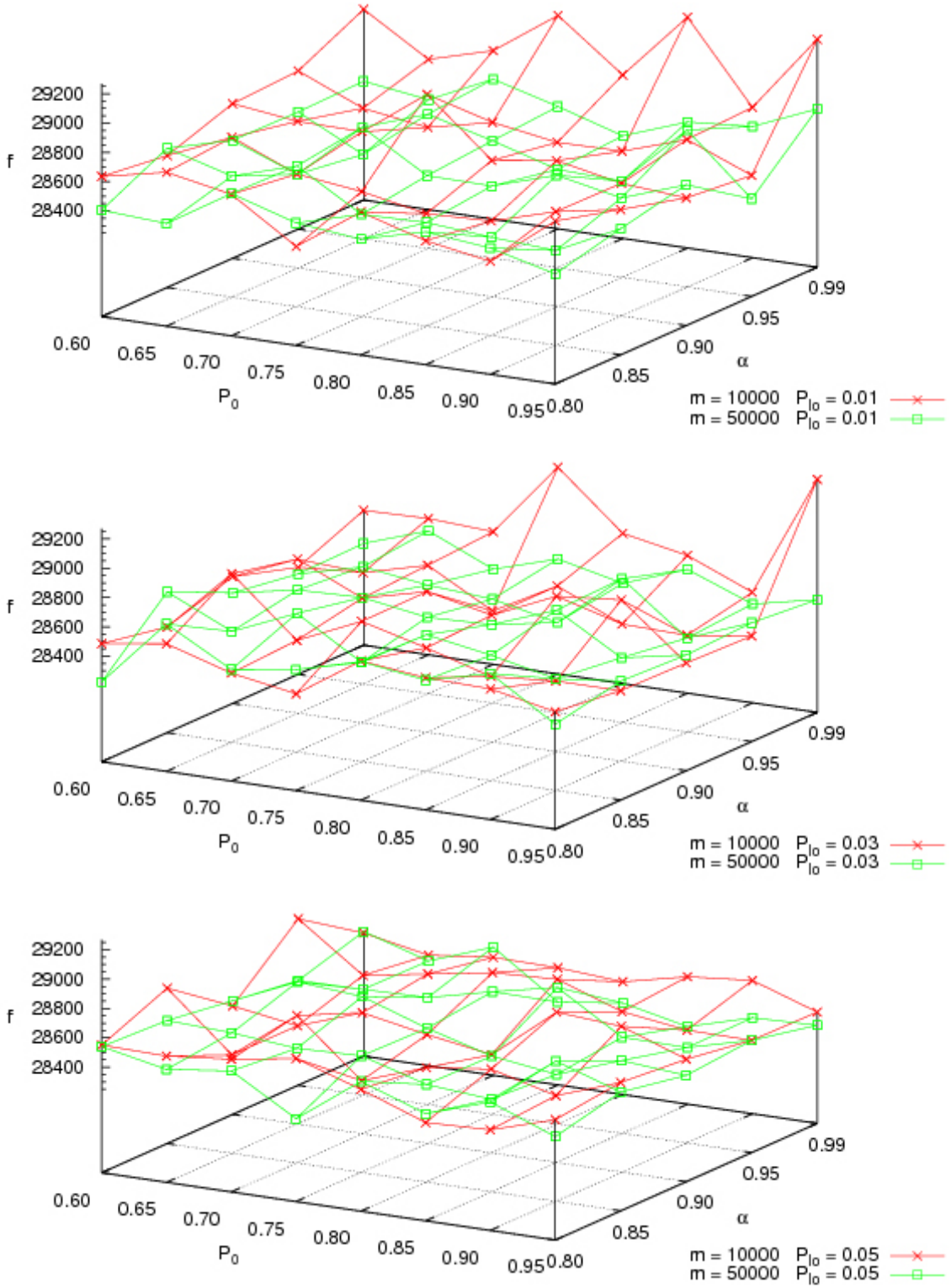
Rys. 3. Zależność jakości wyznaczonego rozwiązania od parametrów α i P_0 dla ustalonej liczby iteracji m oraz prawdopodobieństwa lokalnej optymalizacji P_{lo}. Wyniki dla zestawu danych ESC25

Fig. 3. The solution quality depending on the parameters α i P_0 for the given number of iterations m and probability of local optimization P_0. Results for the data set ESC25



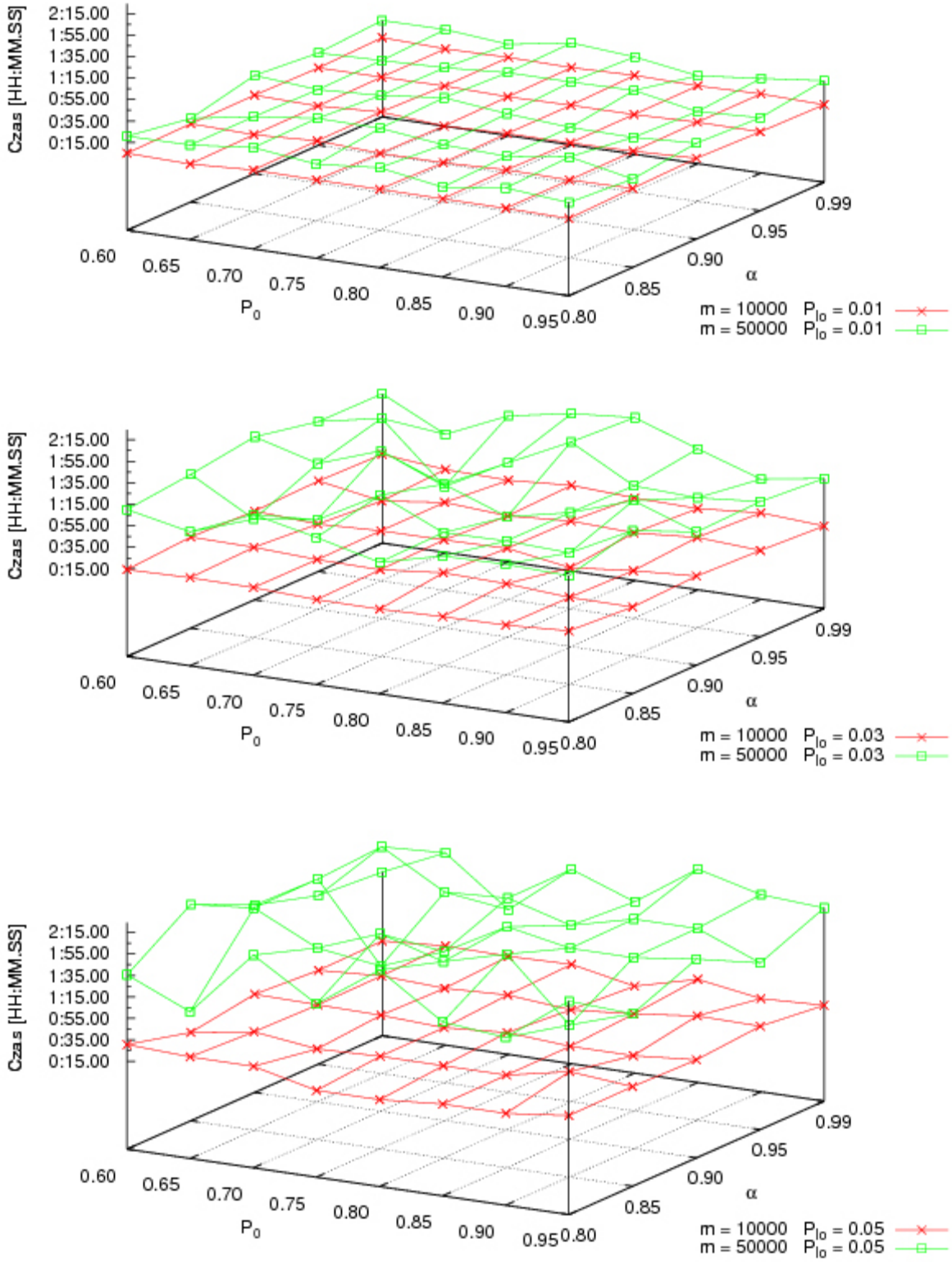
Rys. 4. Zależność czasu wyznaczania rozwiązania od parametrów α i P_0 dla ustalonej liczby iteracji m oraz prawdopodobieństwa lokalnej optymalizacji P_{lo}. Wyniki dla zestawu danych ESC25

Fig. 4. The computation time depending on the parameters α i P_0 for the given number of iterations m and probability of local optimization P_0. Results for the data set ESC25



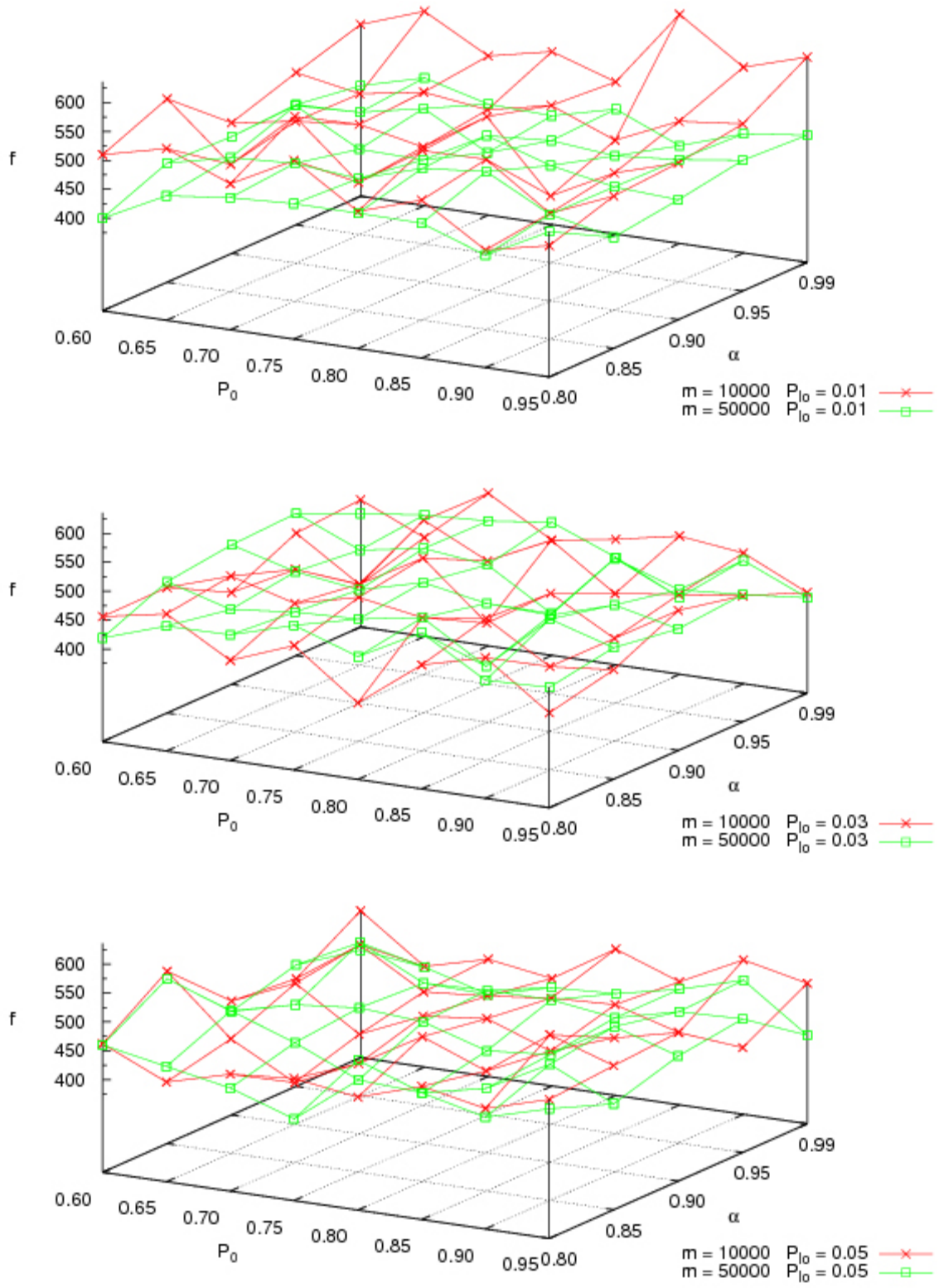
Rys. 5. Zależność jakości wyznaczonego rozwiązania od parametrów α i P_0 dla ustalonej liczby iteracji m oraz prawdopodobieństwa lokalnej optymalizacji P_{lo}. Wyniki dla zestawu danych p43.1

Fig. 5. The solution quality depending on the parameters α i P_0 for the given number of iterations m and probability of local optimization P_0. Results for the data set p43.1



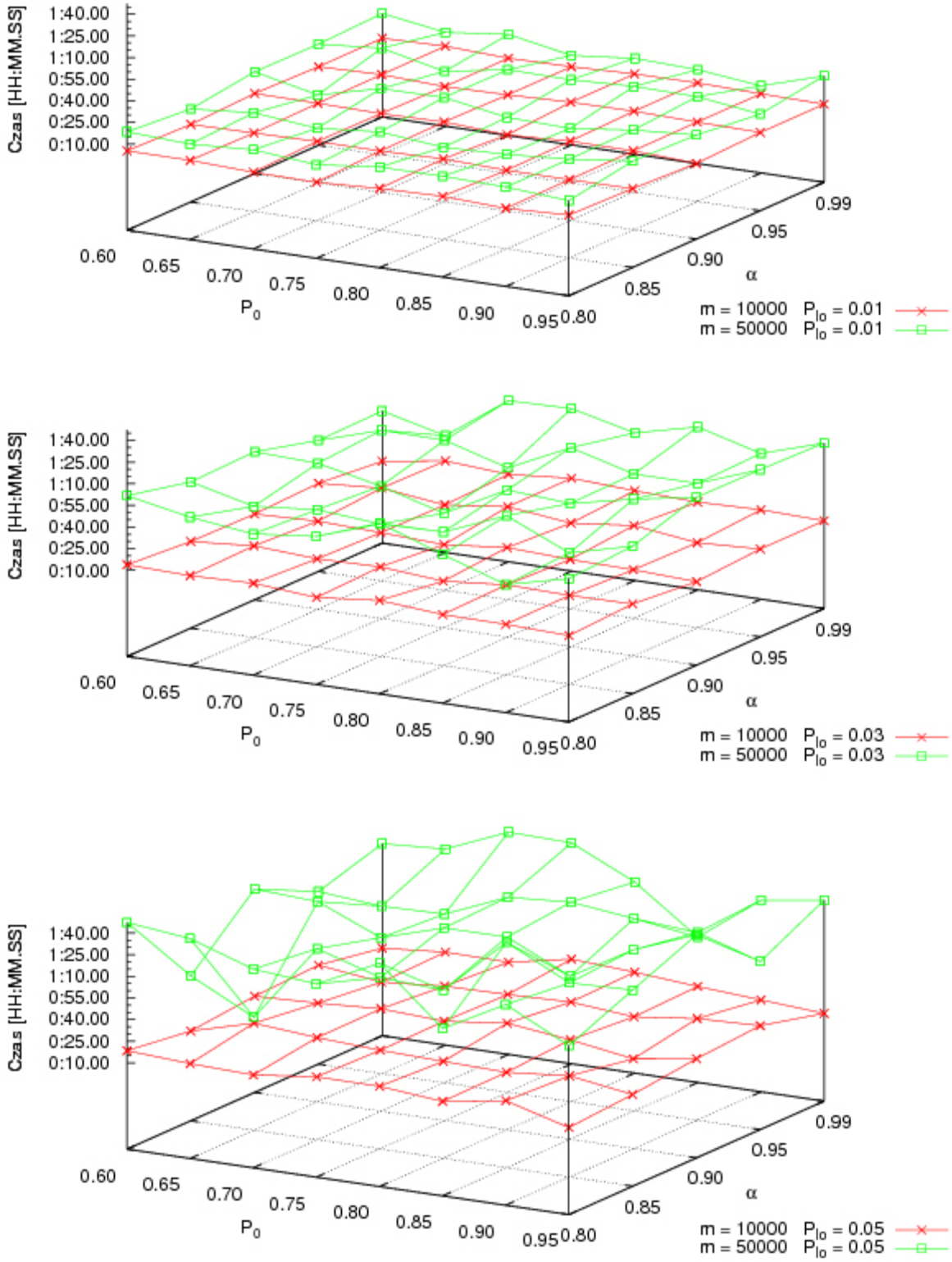
Rys. 6. Zależność czasu wyznaczania rozwiązania od parametrów α i P_0 dla ustalonej liczby iteracji m oraz prawdopodobieństwa lokalnej optymalizacji P_{lo}. Wyniki dla zestawu danych p43.1

Fig. 6. The computation time depending on the parameters α i P_0 for the given number of iterations m and probability of local optimization P_0. Results for the data set p43.1



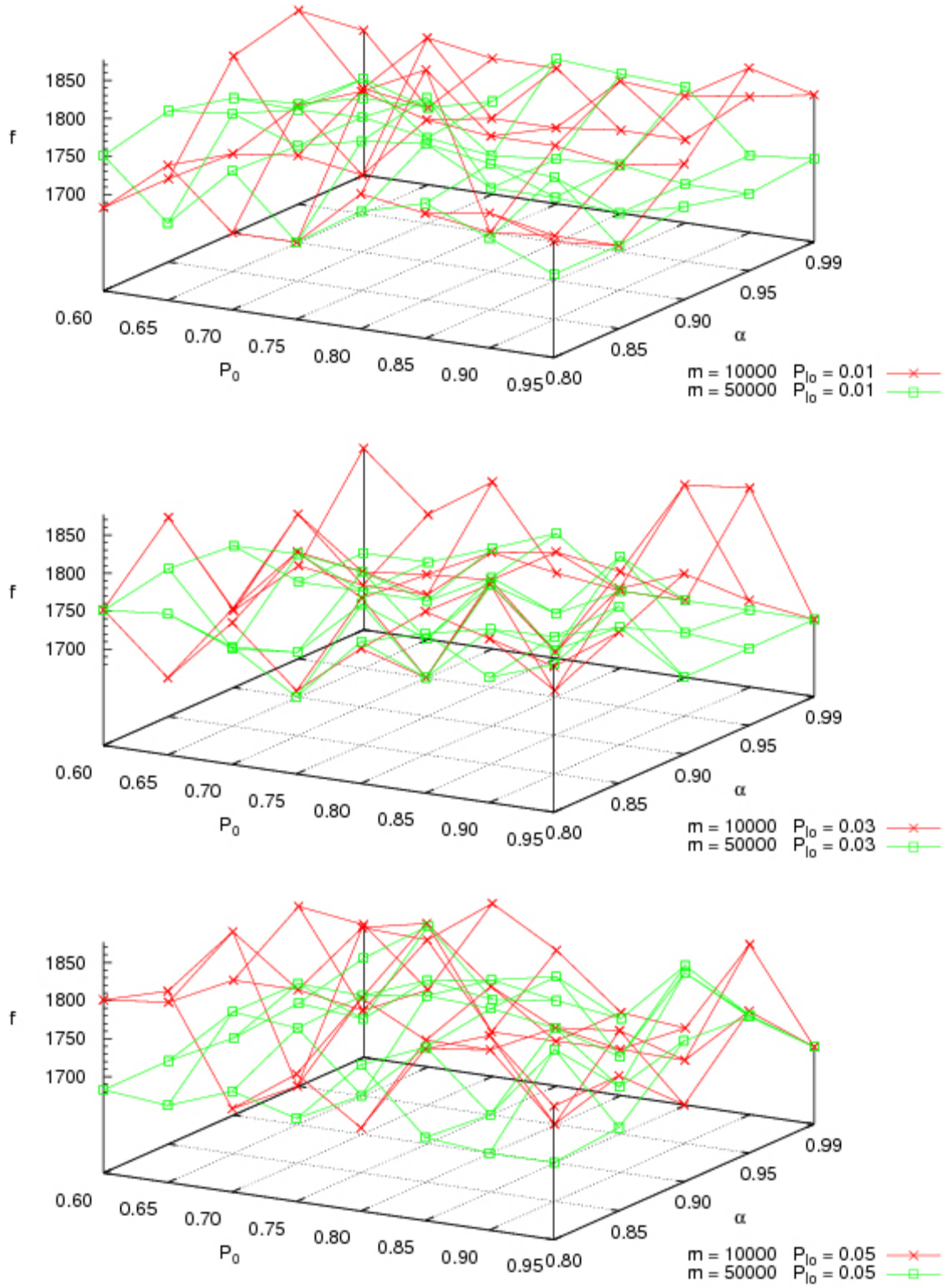
Rys. 7. Zależność jakości wyznaczonego rozwiązania od parametrów α i P_0 dla ustalonej liczby iteracji m oraz prawdopodobieństwa lokalnej optymalizacji P_{lo}. Wyniki dla zestawu danych prob.42

Fig. 7. The solution quality depending on the parameters α i P_0 for the given number of iterations m and probability of local optimization P_0. Results for the data set prob.42



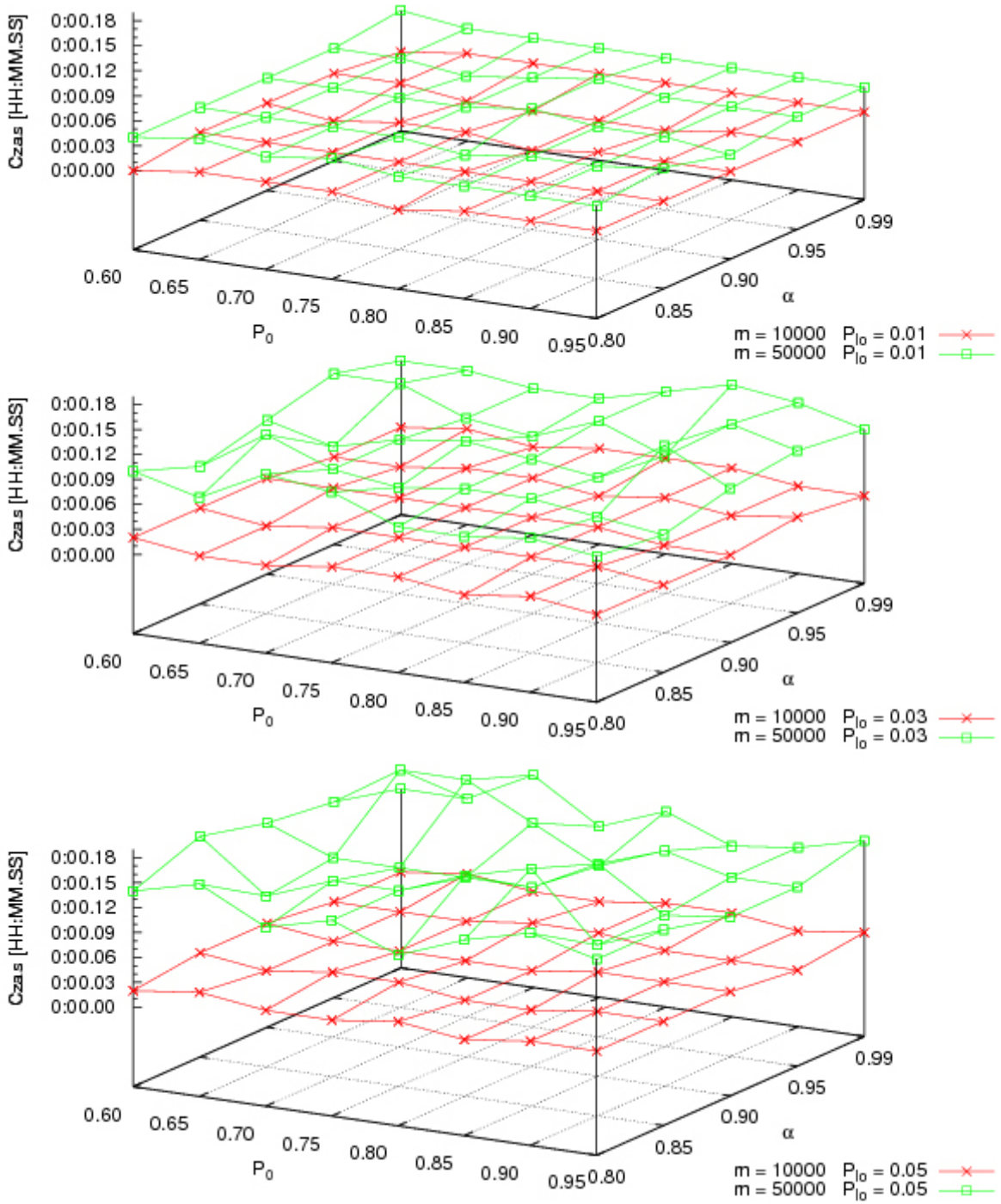
Rys. 8. Zależność czasu wyznaczania rozwiązania od parametrów α i P_0 dla ustalonej liczby iteracji m oraz prawdopodobieństwa lokalnej optymalizacji P_{lo}. Wyniki dla zestawu danych prob.42

Fig. 8. The computation time depending on the parameters α i P_0 for the given number of iterations m and probability of local optimization P_0. Results for the data set prob.42



Rys. 9. Zależność jakości wyznaczonego rozwiązania od parametrów α i P_0 dla ustalonej liczby iteracji m oraz prawdopodobieństwa lokalnej optymalizacji P_{lo}. Wyniki dla zestawu danych ESC12

Fig. 9. The solution quality depending on the parameters α i P_0 for the given number of iterations m and probability of local optimization P_0. Results for the data set ESC12



Rys. 10. Zależność czasu wyznaczania rozwiązania od parametrów α i P_0 dla ustalonej liczby iteracji m oraz prawdopodobieństwa lokalnej optymalizacji P_{lo}. Wyniki dla zestawu danych ESC12

Fig. 10. The computation time depending on the parameters α i P_0 for the given number of iterations m and probability of local optimization P_0. Results for the data set ESC12

Ostatnim zestawem danych użytym w pierwszej części badań był zestaw ESC12, a uzyskane wyniki zostały przedstawione na wykresach na rys. 9 i 10. Dla tego zestawu danych w 52 przypadkach uzyskane zostało optymalne rozwiązanie o jakości $f_{opt} = 1675$ [47]. Czasy minimalny i maksymalny wyznaczania rozwiązania optymalnego były równe odpowiednio 0:00.01 i 0:00.19. Największa liczba optymalnych rozwiązań równa 15 została wyznaczona dla parametrów $m = 50000$, $P_{lo} = 0.03$. Natomiast najmniejszą liczbę optymalnych rozwiązań, równą 5, wyznaczono dla wartości parametrów $m = 10000$, $P_{lo} = 0.01$. Najgorszym wyznaczonym rozwiązaniem było rozwiązanie o jakości $f = 1876$, gorsze o ok. 12% od rozwiązania optymalnego. Rozwiązanie to uzyskano dla wartości parametrów: $\alpha = 0.85$, $P_0 = 0.80$, $m = 10000$, $P_{lo} = 0.01$, a czas jego wyznaczania był równy 0:00.01.

W drugiej części badań, na podstawie wyników uzyskanych w pierwszej części, dokonano wyboru wartości parametrów algorytmu oraz przeprowadzone zostały badania dla większej liczby zestawów danych. Wybrane zostały wartości parametrów, dla których uzyskane zostało najlepsze rozwiązanie dla zestawów danych użytych w pierwszej części badań. Wykonano serie badań dla następujących wartości parametrów:

- eksperyment 1: $\alpha = 0.80$, $P_0 = 0.80$, $m = 10000$, $P_{lo} = 0.03$,
- eksperyment 2: $\alpha = 0.80$, $P_0 = 0.60$, $m = 50000$, $P_{lo} = 0.03$,
- eksperyment 3: $\alpha = 0.95$, $P_0 = 0.70$, $m = 50000$, $P_{lo} = 0.05$.

Tabela 4

Jakość wyznaczonego rozwiązania z użyciem algorytmów *MS_H_SOP*, *SD_LS* i *G_LS*

Zestaw danych	n	f_{opt}	*MS_H_SOP*			*SD_LS*	*G_LS*
			eksperyment 1	eksperyment 2	eksperyment 3		
ESC07	9	2125	2125	2125	2125	2550	2125
ESC11	13	2075	2145	2129	2150	2376	2684
ESC12	14	1675	1762	1797	1738	1752	1839
ESC25	27	1681	2889	2581	2140	3043	4466
ESC47	49	1288	3748	3240	3583	5357	4357
br17.10	18	55	55	58	55	58	60
br17.12	18	55	55	55	57	58	60
prob.42	42	243	372	453	444	469	572
p43.1	44	27990	28450	28225	28555	28565	55185
ry48p.1	49	15805	17425	18853	18664	21373	21711
ft53.1	54	7531	10297	10269	9374	10425	11810
ft70.1	71	39313	44216	43915	43623	45345	45592

Dodatkowo przeprowadzono badania z użyciem algorytmu przeszukiwania lokalnego w wersji stromej (ang. *steepest descent local search*) i zachłannej (ang. *greedy local search*) [38, 41]. Wyniki przeprowadzonych badań przedstawione zostały w tabelach 4 i 5. W tabelach algorytmy lokalnej optymalizacji w wersji stromej i zachłannej zaznaczono odpowiednio jako *SD_LS* i *G_LS*, a algorytm przedstawiony w rozdziale 3 oznaczono jako *MS_H_SOP*.

Dla każdego zestawu danych zamieszczono informację o liczbie miast (kolumna *n*) oraz na podstawie [47] informację o jakości znanego rozwiązania optymalnego (kolumna f_{opt}).

Tabela 5

Czas wyznaczania rozwiązania z użyciem algorytmów *MS_H_SOP*, *SD_LS* i *G_LS*

Zestaw danych	*n*	f_{opt}	*MS_H_SOP*			*SD_LS*	*G_LS*
			eksperyment 1	eksperyment 2	eksperyment 3		
ESC07	9	2125	0:00.00	0:00.02	0:00.02	0:00.00	0:00.00
ESC11	13	2075	0:00.01	0:00.08	0:00.09	0:00.00	0:00.00
ESC12	14	1675	0:00.02	0:00.09	0:00.12	0:00.00	0:00.00
ESC25	27	1681	0:01.16	0:05.19	0:08.29	0:00.05	0:00.03
ESC47	49	1288	0:34.31	2:23.02	3:58.08	0:03.29	0:03.11
br17.10	18	55	0:00.07	0:00.22	0:00.53	0:00.01	0:00.00
br17.12	18	55	0:00.08	0:00.25	0:00.38	0:00.03	0:00.02
prob.42	42	243	0:15.02	0:41.04	1:06.20	0:01.27	0:00.54
p43.1	44	27990	0:13.36	1:10.06	2:13.18	0:00.25	0:00.23
ry48p.1	49	15805	0:31.04	2:15.24	3:25.30	0:03.28	0:03.11
ft53.1	54	7531	0:45.38	3:52.11	5:47.15	0:08.55	0:05.57
ft70.1	71	39313	14:31.14	16:02.29	19:52.33	3:22.32	7:47.57

Dla trzech zestawów danych (ESC07, br17.10 i br17.12) wyznaczone zostało optymalne rozwiązanie z użyciem algorytmu *MS_H_SOP*, przy czym tylko dla jednego zestawu (ESC07) zostało ono wyznaczone we wszystkich trzech przeprowadzonych eksperymentach, a dla dwóch pozostałych zestawów rozwiązanie optymalne zostało wyznaczone w dwóch eksperymentach. Tylko w jednym przypadku (zestaw ESC07) wyznaczono optymalne rozwiązanie z użyciem algorytmu *G_LS*, a żadnego optymalnego rozwiązania nie udało się wyznaczyć z użyciem algorytmu *SD_LS*. W pozostałych przypadkach rozwiązania wyznaczone za pomocą obydwu algorytmów przeszukiwania lokalnego były gorsze od rozwiązań wyznaczonych algorytmem *MS_H_SOP*. Algorytmy przeszukiwania lokalnego mają natomiast przewagę nad algorytmem *MS_H_SOP* ze względu na czas wyznaczania rozwiązania. W każdym przeprowadzonym eksperymencie z użyciem algorytmu *MS_H_SOP* czas ten był większy niż czas wyznaczania rozwiązań z użyciem algorytmów przeszukiwania lokalnego.

5. Podsumowanie

W pracy zaproponowany został hybrydowy algorytm wielokrotnego startu rozwiązywania problemu SOP. Algorytm ten jest połączeniem algorytmu symulowanego wyżarzania i lokalnej optymalizacji. Dla każdego startu algorytmu jest generowane nowe rozwiązanie początkowe, przy czym jest ono różne od rozwiązań początkowych wygenerowanych w poprzednich startach algorytmu, a celem jest przeniesienie procesu wyszukiwania w inny obszar przestrzeni rozwiązań. Rozwiązanie startowe generowane jest w sposób losowy, przy

czym losowanie odbywa się tak, aby w wygenerowanym rozwiązaniu spełnione były relacje pierwszeństwa.

Dla opracowanego algorytmu przeprowadzone zostały badania eksperymentalne z użyciem danych pochodzących z biblioteki TSPLIB. Celem badań było określenie wpływu parametrów sterujących algorytmu na jakość wygenerowanego rozwiązania i czas jego wyznaczania. Zbadane zostały następujące parametry: współczynnik redukcji temperatury α, początkowa wartość prawdopodobieństwa P_0 akceptacji gorszego rozwiązania, liczba wykonanych iteracji m podczas pojedynczego wykonania algorytmu oraz wartość prawdopodobieństwa P_{lo} lokalnej optymalizacji. Dodatkowo przeprowadzono badania z użyciem algorytmu przeszukiwania lokalnego w wersji stromej i zachłannej, a uzyskane wyniki porównano z wynikami dla algorytmu hybrydowego.

BIBLIOGRAFIA

1. Aarts E., Korst J.: Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing. John Wiley & Sons, Inc. New York, NY, USA 1989.
2. Aarts E., Korst J., Michiels W.: Simulated Annealing. [in:] Search Methodologies Introductory Tutorials in Optimization and Decision Support Technique, Springer, New York, NY, USA 2000, p. 187–210.
3. Alcaide D., Rodriguez–Gonzalez A., Sicilia J.: An approach to solve a hierarchical stochastic sequential ordering problem. Omega, Vol. 31, Issue 3, 2003, p. 169–187.
4. Alonso–Ayuso A., Detti P., Escudero L. F., Ortuño M. T.: On Dual Based Lower Bounds for the Sequential Ordering Problem with Precedences and Due Dates. Annals of Operations Research, Vol. 124, Issue 1–4, 2003, p. 111–131.
5. Anghinolfi D., Montemanni R., Paolucci M., Gambardella L. M.: A hybrid particle swarm optimization approach for the sequential ordering problem. Computers & Operations Research, Vol. 38, Issue 7, 2011, p. 1076–1085.
6. Bock F.: An algorithm for solving traveling–salesman and related network optimization problems. Research Report, Armour Research Foundation, Presented at the Operations Research Society of America Fourteenth National Meeting, St. Louis, October 24, 1958.
7. Cerny V.: A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications, Vol. 45, No. 1, 1985, p. 41–55.
8. Cormen T. H., Leiserson C. E., Rivest R. L.: Wprowadzenie do algorytmów. WNT, Warszawa 2007.

9. Dantzig G. B., Fulkerson R., Johnson S. M.: Solution of a large–scale traveling salesman problem. Operations Research, Vol. 2, Issue 4, 1954, p. 393–410.
10. Escudero L. F.: An inexact algorithm for the sequential ordering problem. European Journal of Operational Research, Vol. 37, Issue 2, 1988, p. 236–249.
11. Escudero L. F., Guignard M., Malik K.: A Lagrangian relax–and–cut approach for the sequential ordering problem with precedence relationships. Annals of Operations Research, Vol. 50, No. 1, 1994, p. 219–237.
12. Escudero L. F., Ortuño M. T.: On Due-Date Based Valid Cuts for the Sequential Ordering Problem. Top, Vol. 5, Issue 1, 1997, p. 159–166.
13. Escudero L. F., Sciomachen A.: Local search procedures for improving feasible solutions to the sequential ordering problem. Annals of Operations Research, Vol. 43, No. 7, 1993, p. 397–416.
14. Gambardella L. M., Dorigo M.: An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem. INFORMS Journal on Computing, Vol. 12, No. 2, 2000, p. 237–255.
15. Gambardella L. M., Montemanni R., Weyland D.: An Enhanced Ant Colony System for the Sequential Ordering Problem. Operations Research Proceedings 2012, p. 355–360.
16. Grötschel M., Padberg M. W.: On the symmetric travelling salesman problem I: Inequalities. Mathematical Programming, Vol. 16, 1979, p. 265-280.
17. Guerriero F., Mancini M.: A cooperative parallel rollout algorithm for the sequential ordering problem. Parallel Computing, Vol. 29, No. 5, 2003, p. 663–677.
18. Hernŕdvölgyi I. T.: Solving the sequential ordering problem with automatically generated lower bounds. [in:] Proceedings of Operations Research, Springer–Verlag Berlin, Heidelberg, Germany 2003, p. 355–362.
19. Homaifar A., Guan S., Liepins G. E.: A New Approach on the Traveling Salesman Problem by Genetic Algorithms. In Proceedings of the Fifth International Conference on Genetic Algorithms, 1993, p. 460–466.
20. Johnson D. S., McGeoch, L. A.: The Traveling Salesman Problem: A Case Study in Local Optimization. [in:] Local Search in Combinatorial Optimisation, John Wiley and Sons Ltd, London, UK 1997, p. 215–310.
21. Jungnickel D.: Graphs, networks and algorithms. Springer, Berlin, Germany 1999.
22. Kirkpatrick S.: Optimization by Simulated Annealing: Quantitative Studies. Journal of Statistical Physics, Vol. 34, Issue 5–6, 1984, p. 975–986.
23. Kirkpatrick S., Gelatt C. D., Vecchi M. P.: Optimization by simulated annealing. Science, Vol. 220, No. 4598, 1983, p. 671–680.
24. Landau L. D., Lifszyc J. M.: Fizyka statystyczna część 1. Wydawnictwo Naukowe PWN, Warszawa 2012.

25. Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G., Shmoys D. B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley & Sons, 1985.
26. Lenstra J. K., Rinnooy Kan A. H. G.: Some Simple Applications of the Travelling Salesman Problem. Operational Research Quarterly, Vol. 26, No. 4, Part 1, 1975, p. 717–733.
27. Lin S.: Computer solutions of the traveling salesman problem. Bell System Technical Journal, Vol. 44, 1965, p. 2245–2269.
28. Lin S., Kernighan B. W.: An Effective Heuristic Algorithm for the Traveling–Salesman Problem. Operations Research, Vol. 21, No. 2, 1973, p. 498–516.
29. Little J. D. C., Murty K. G., Sweeney D. W., Karel C.: An algorithm for the traveling salesman problem. Operations Research, Vol. 11, 1963, p. 972–989.
30. Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Teller E.: Equation of state calculation by fast computing machines. The Journal of Chemical Physics, Vol. 21, No. 6, 1953, p. 1087–1091.
31. Metropolis N., Ulam S.: The Monte Carlo Method. Journal of the American Statistical Association, Vol. 44, No. 247, 1949, p. 335–341.
32. Michalewicz Z., Fogel D. B.: How to Solve It: Modern Heuristics, Springer, New York, NY, USA 2000.
33. Michalewicz Z., Fogel D. B.: Jak to rozwiązać czyli nowoczesna heurystyka. WNT, Warszawa 2006.
34. Michalewicz Z.: Algorytmy genetyczne + struktury danych = programy ewolucyjne. WNT, Warszawa 1996.
35. Mojana M., Montemanni R., Di Caro G., Gambardella L. M.: A branch and bound approach for the sequential ordering problem. Lecture Notes in Management Science, Vol. 4, 2012, p. 266–273.
36. Montemanni R., Smith D. H., Gambardella L. M.: A heuristic manipulation technique for the sequential ordering problem. Computers & Operations Research, Vol. 35, Issue 12, 2008, p. 3931–3944.
37. Montemanni R., Smith D. H., Gambardella L. M.: Ant Colony Systems for Large Sequential Ordering Problems. [in:] Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007), 2007, p. 60–67.
38. Osman I. H., Kelly J. P.: Meta-Heuristics: An Overview. [in:] Meta-heuristics: Theory and Applications. Kluwer Academic Publishers, 1996.
39. Padberg M. W., Rao M. R.: The travelling salesman problem and a class of polyhedra of diameter two. Mathematical Programming, Vol. 7, Issue 1, 1974, p. 32–45.
40. Papadimitriou C. H., Steiglitz K.: On the Complexity of Local Search for the Traveling Salesman Problem. SIAM Journal on Computing, Vol. 6, 1977, p. 76–83.

41. Rayward–Smith V. J., Osman I. H., Reeves C. R., Smith G. D.: Modern Heuristic Search Methods. John Wiley & Sons Ltd., Chichester, England 1996.
42. Reeves C. R.: Modern heuristic techniques for combinatorial problems. John Wiley & Sons, Inc. New York, NY, USA 1993.
43. Reinelt G.: The Traveling Salesman: computational solutions for TSP applications. Springer–Verlag Berlin, Heidelberg, Germany 1994.
44. Sanvicente–Sánchez H., Frausto–Solís J.: MPSA: A Methodology to Parallelize Simulated Annealing and It s Application to the Traveling Salesman Problem. MICAI 2002, LNAI 2313, 2002, p. 89–97.
45. Seo D., Moon B.: A hybrid genetic algorithm based on complete graph representation for the sequential ordering problem. LNCS, Vol. 2723, 2003, p. 669–680.
46. Yang R.: Solving Large Traveling Salesman Problems with Small Populations. Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA 97 Second International Conference On (Conf. Publ. No. 446), 1997, p. 157–162.
47. TSPLIB, a library of sample instances for the TSP (and related problems) from various sources and of various types: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Abstract

Traveling salesman problem (TSP) is one of studied problems in combinatorial optimization area. In TSP a set of n cities is given and for each pair of cities (i, j) a distance $d(i, j)$ is defined. The goal of the problem is to find a minimum length route of a salesman who visiting each city exactly once and returning at the end to the initial city. TSP is symmetric, i.e. for each pair of cities (i, j) the distances satisfy $d(i, j) = d(j, i)$. The problem can be modeled as an undirected weighted graph G where cites are represented by vertices and distance between cites i and j is represented by weight of edge (i, j). In terms of graph theory the goal of TSP is to find a minimum weight Hamiltonian cycle in a undirected weighted graph. There is also an asymmetric traveling salesman problem (ATSP) where the distances $d(i, j)$ and $d(j, i)$ may be different and it is modeled as a directed weighted graph. Sequential ordering problem (SOP) is a related problem to ATSP. The goal of SOP is to find a minimum length Hamiltonian path satisfying precedence relationships among the vertices.

In the paper, a multistart hybrid algorithm to solving SOP is presented. The algorithm based on simulated annealing algorithm and local optimization method. For each start of the algorithm a new initial solution is generated but it is different from the initial solutions generated in the previous starts. It guarantees that algorithm starts to search in different space of solutions. The initial solution is random generated but it satisfying precedence relationships

among the vertices. The algorithm was tested using a set of test problems taken from the TSPLIB library [47]. The aim of tests was to investigate an impact of the control algorithm parameters on the computation time and the solution quality.

Adresy

Jacek WIDUCH: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, jacek.widuch@polsl.pl.
Artur KLYTA: artur.klyta@gmail.com.