ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 123 Nr kol. 1389

Jacek JAGIEŁŁO
Instytut Informatyki Uniwersytetu Wrocławskiego

WYKORZYSTANIE ZALEŻNOŚCI RUCHÓW DO PRZYSPIESZENIA POSZUKIWANIA LOKALNEGO I POSZUKIWANIA Z TABU

Streszczenie. Poszukiwanie lokalne (ang. *local search*) i poszukiwanie z tabu (ang. *tabu search*) są często dziś stosowanymi heurystykami do rozwiązywania trudnych praktycznych problemów optymalizacji dyskretnej. Przedstawiona w pracy metoda przyspieszenia działania algorytmów lokalnego poszukiwania i poszukiwania z tabu pozwala skrócić czas szukania rozwiązania lub poprawić jakość znalezionego rozwiązania bez wydłużania czasu obliczeń.

DEPENDENT MOVES IN LOCAL AND TABU SEARCH

Summary. Local search and tabu search are common used heuristics in solving of hard practical optimization problems. Presented method, in some cases, allow to achieve shorter time of searching desired solution or allow to find better solution in the same time.

1. Sformułowanie problemu

W skończonej przestrzeni rozwiązań V, mając daną funkcję celu $f : V \to \mathcal{R}$, szukamy jej globalnego maksimum, tzn. takiego $v^* \in V$, że dla każdego $v \in V$ zachodzi $f(v^*) \geq f(v)$ (szukanie globalnego minimum funkcji celu f jest równoważne szukaniu globalnego maksimum funkcji $-f$). Innym wariantem tego problemu jest szukanie takiego v, że $f(v) \geq p$, dla zadanej wartości progowej p. Ogólnie o tego typu wariantach będziemy mówić jako o szukaniu zadowalającego rozwiązania.

Przykładem tego rodzaju problemu może być **zadanie przydziału prac** (ang. *assigment problem*). Mając dane dla każdej z n prac i każdego z n wykonawców zysk z wykonania danej pracy przez danego wykonawcę należy znaleźć taki przydział prac wykonawcom, aby zmaksymalizować sumaryczny zysk. Czyli dla danej macierzy zysków $\{a_{i,j}\}_{n \times n}$ należy znaleźć w przestrzeni permutacji n-elementowych ($V = \Pi_n$) maksimum globalne funkcji celu, czyli takie π^*, że

$$f(\pi^*) = \max_{\pi=(\pi(1),\ldots,\pi(n))\in\Pi_n} \sum_{i=1}^{n} a_{i,\pi(i)}.$$

Rozwiązywanie powyższego przykładu metodą lokalnego przeszukiwania będzie miało jedynie znaczenie ilustracyjne, bowiem jest to problem posiadający rozwiązanie o wielomianowej złożoności. Na tym prostym przykładzie zostanie zaprezentowane pojęcie ruchów lokalnych i zależności ruchów lokalnych.

Problemem, którego rozwiązanie prócz ilustracyjnego, ma praktyczne znaczenie, jest problem znalezienia dostatecznie gęstego krawędziowo podgrafu o zadanej liczbie wierzchołków. Dokładniej, dla zadanego n-wierzchołkowego grafu nieskierowanego G, dla zadanej liczby m – liczby wierzchołków podgrafu ($0 \leq m \leq n$), dla zadanej liczby p – progu dla liczby krawędzi w podgrafie, szukamy takiego m-wierzchołkowego grafu H – podgrafu grafu G, że liczba krawędzi H jest nie mniejsza niż p. Wartością funkcji celu jest liczba krawędzi podgrafu. Problem ten będziemy nazywać **szukaniem gęstego podgrafu**.

1.1. Lokalne poszukiwanie

Jedną ze stosowanych strategii znajdowania zadowalającego rozwiązania jest metoda lokalnego poszukiwania (ang. *local search*) przedstawiona poniżej:

ALGORYTM LP (Lokalne Poszukiwanie)

1. wybierz punkt startowy v {losowo lub według pewnej reguły};
2. repeat
3. wybierz punkt w z sąsiedztwa v taki, że $f(w) > f(v)$;
4. zastąp punkt v punktem w;
5. until punkt v będzie lokalnym względem sąsiedztwa maksimum f

{czyli dla każdego w z sąsiedztwa v będzie zachodziło $f(v) \geq f(w)$};

Opisane przez pętlę algorytmu postępowanie nazywać będziemy szukaniem końca ścieżki polepszającej. Dla każdego punktu na ścieżce (z wyjątkiem ostatniego) kolejny wybrany punkt będziemy nazywać dominatorem poprzedniego (bo dominuje on nad nim wartością funkcji celu). Ostatni punkt ścieżki polepszającej niezdominowany przez żaden wierzchołek z sąsiedztwa nazywać będziemy lokalnym maksimum lub hegemonem.

Aby zastosować przedstawiony algorytm do problemu optymalizacji dyskretnej, należy zdefiniować sąsiedztwo punktu. Ogólnie, sąsiedztwo jest definiowane jako pewna funkcja $N : V \to 2^V$, czyli $N(v)$ jest zbiorem sąsiadów punktu $v \in V$.

W zadaniu przydziału prac sąsiedztwo może być zdefiniowane na wiele sposobów. Na przykład sąsiadem wierzchołka (punktu w przestrzeni rozwiązań – w przestrzeni permutacji) może być permutacja różniąca się od wyjściowej o pojedynczą transpozycję dwóch sąsiednich elementów (oznaczoną $t_{i,i+1}$). Czyli $N(\pi) = \{t_{i,i+1}(\pi) : i = 1, \ldots, n-1\}$.

Innym sąsiedztwem może być zbiór permutacji różniących się o dowolną pojedynczą transpozycję $t_{i,j}$, gdzie $i, j = 1, \ldots, n$.

W zadaniu szukania gęstego podgrafu przestrzenią rozwiązań może być zbiór m-elementowych podzbiorów zbioru wierzchołków grafu G. Rozwiązaniem sąsiednim dla danego podzbioru wierzchołków grafu może być podzbiór m-elementowy, który różni się od zbioru wyjściowego jednym elementem. Inaczej mówiąc – przekrój danego podzbioru z jego sąsiadem jest zawsze zbiorem $(m-1)$-elementowym.

Trzeci wiersz algorytmu szukania końca ścieżki polepszającej bywa realizowany na kilka sposobów, np. poprzez losowanie sąsiada aż do momentu, gdy zostanie znaleziony lepszy, systematyczne przeglądanie sąsiadów aż zostanie znaleziony lepszy, systematyczne przejrzenie wszystkich sąsiadów i wybór najlepszego z nich itp.

Jednorazowe szukanie końca ścieżki polepszającej może nie dać zamierzonego efektu. Zwykle stosuje się wtedy wielokrotne powtarzanie szukania i wybór najlepszego ze znalezionych optimów.

1.2. Jednorodna funkcja sąsiedztwa

W związku z tym, że dla wielu problemów optymalizacyjnych opis punktu ma złożoną strukturę (jest np. ciągiem decyzji, permutacją, itp.), sąsiada definiuje się jako punkt, którego opis różni się w niewielkim stopniu od punktu wyjściowego (np. zmianą jednego elementu w ciągu decyzji, pojedynczą transpozycją itp.). Zwykle pociąga to za sobą niewielką zmianę przebiegu obliczania funkcji celu przy przechodzeniu od punktu do jego sąsiada i umożliwia nieliczenie funkcji celu od początku.

Przy tak zdefiniowanym sąsiedztwie każdy punkt ma taką samą liczbę sąsiadów (oznaczmy ją k) i dla każdego punktu sąsiedzi są konstruowani według jednej metody. Tak zachowującą się funkcję sąsiedztwa nazywać będziemy **jednorodną funkcją sąsiedztwa**.

W przypadku jednorodnej funkcji sąsiedztwa sąsiedztwo można zdefiniować mniej ogólnie – globalnie dla całej przestrzeni rozwiązań. **Ruchem** będziemy nazywać dowolną funkcję o argumentach i wartościach z przestrzeni rozwiązań. Część ruchów wyróżnionych poprzez zastosowanie ich do konstruowania sąsiedztwa będziemy nazywać **ruchami lokalnymi**.

Niech $\hat{R} = \{\hat{r}_i : i = 1, \ldots, k\}$ będzie zbiorem interesujących nas k ruchów lokalnych postaci: $\hat{r}_i : V \to V$ dla $i = 1, \ldots, k$. Wówczas dla każdego punktu $v \in V$ sąsiedztwo można zdefiniować jako

$$N(v) = \{\hat{r}_i(v) : i = 1, \ldots, k\} = \{r(v) : r \in \hat{R}\}$$

Definicja: Funkcję sąsiedztwa N nazywamy **jednorodną**, jeśli istnieje $k \in \mathcal{N}$ oraz ruchy $r_1, \ldots, r_k$ takie, że dla każdego $v \in V$ mamy $N(v) = \{r_1(v), \ldots, r_k(v)\}$.

W przypadku jednorodnej funkcji sąsiedztwa zamiast o sąsiadach mówić możemy o lokalnych ruchach, które do tych sąsiadów prowadzą.

Zadanie przydziału prac z przestrzenią rozwiązań będącą przestrzenią n-elementowych permutacji ma w naturalny sposób jednorodną funkcję sąsiedztwa, bo transpozycje są naturalnymi ruchami lokalnymi. W przypadku transpozycji elementów sąsiednich ruchy lokalne definiujemy jako $\hat{r}_i = t_{i,i+1}$ dla $i = 1, \ldots, k$, gdzie $k = n - 1$.

Dla zadania szukania gęstego podgrafu podana przykładowa funkcja sąsiedztwa jest także jednorodna, bo za k wystarczy przyjąć $m(n-m)$, zaś ruchy lokalne można zdefiniować następująco. Mając ponumerowane wierzchołki grafu G, dla dowolnego podgrafu H, dla $1 \leq i \leq n$ oraz $1 \leq j \leq n - m$ określamy, że $\hat{r}_{(i-1)(n-m)+j}(H) = H'$, gdzie podgraf H' powstaje z H poprzez usunięcie i-tego (według numeracji) wierzchołka z H i wstawienie j-tego (według numeracji) wierzchołka z dopełnienia podgrafu H wraz z odpowiadającymi im krawędziami. Numer ruchu lokalnego jednoznacznie wskazuje, który wierzchołek ma być usunięty, a który wstawiony podczas wykonywania tego ruchu.

2. Funkcyjna niezależność ruchów w wariancie najlepszego sąsiada

Najprostsza realizacja wyboru najlepszego sąsiada punktu polega na obliczeniu wartości funkcji celu dla wszystkich sąsiadów tego punktu. Można oczywiście nie liczyć dla każdego z nich wartości funkcji od początku, tylko skorzystać z już wykonanych obliczeń. Niezależnie od takiego usprawnienia proponujemy metodę ograniczenia liczby sąsiadów, dla których jest liczona funkcja celu podczas szukania najlepszego sąsiada.

2.1. Funkcyjna niezależność ruchów

Wprowadźmy następujące rozróżnienie konkretnych ruchów od zmiennych przybierających wartości konkretnych ruchów lokalnych. Przez $\hat{r}_i$ oznaczmy i-ty ruch lokalny ze

zbioru $\hat{R}$ wszystkich ruchów lokalnych, zaś przez r_i zmienne, przybierające pewne wartości ze zbioru ruchów R.

Definicja. Dla danych (V, f, R), gdzie V to przestrzeń rozwiązań, f to funkcja celu, zaś R to zbiór ruchów: (ruch $r_1 \in R$ **nazywamy funkcyjnie niezależnym od** $r_2 \in R$)

$$\stackrel{def}{\Longleftrightarrow} \forall v \in V : f(r_1(v)) - f(v) = f(r_1(r_2(v))) - f(r_2(v)).$$

Oznacza to, że zmiana wartości funkcji na skutek ruchu r_1 jest taka sama dla punktu v jak i dla punktu $r_2(v)$.

Twierdzenie. Jeśli zachodzi zależność $f(r_1(v)) \geq f(r_2(v)) \geq \ldots \geq f(r_k(v))$, to dla $1 \leq l \leq k$ i $w = r_l(v)$ zachodzi także $f(r_{i_1}(w)) \geq f(r_{i_2}(w)) \geq \ldots \geq f(r_{i_m}(w))$, gdzie $r_{i_1}, \ldots, r_{i_m}$ są ruchami funkcyjnie niezależnymi od r_l, a ciąg $\{i_j\}_{j=1,\ldots,m}$ jest rosnącym podciągiem ciągu $1, \ldots, n$.

Dowód: Zgodnie z założeniami twierdzenia niech $r_{i_1}, \ldots, r_{i_m}$ będą ruchami funkcyjnie niezależnymi od r_l oraz niech dla każdego $1 \leq i \leq j \leq k$ zachodzi $f(r_i(v)) \geq f(r_j(v))$. Weźmy dowolne a, b takie, że $1 \leq i_a \leq i_b \leq m$. Wówczas:

$$f(r_{i_a}(w)) - f(w) = f(r_{i_a}(r_l(v))) - f(r_l(v)).$$

Korzystając z funkcyjnej niezależności r_{i_a} od r_l otrzymamy dalej

$$f(r_{i_a}(r_l(v))) - f(r_l(v)) = f(r_{i_a}(v)) - f(v)$$

Wykorzystując założenia dotyczące monotoniczności ciągu $\{f(r_i(v))\}_{i=1}^{k}$, otrzymamy:

$$f(r_{i_a}(v)) - f(v) \geq f(r_{i_b}(v)) - f(v),$$

gdyż $i_a \leq i_b$. Przekształcając prawą stronę powyższej nierówności z wykorzystaniem funkcyjnej niezależności ruchu r_{i_b} od ruchu r_l otrzymamy:

$$f(r_{i_b}(v)) - f(v) = f(r_{i_b}(r_l(v))) - f(r_l(v)) = f(r_{i_b}(w)) - f(w)$$

Łącząc wszystkie powyższe równania i nierówności po wykorzystaniu przechodniości, otrzymamy:

$$f(r_{i_a}(w)) - f(w) \geq f(r_{i_b}(w)) - f(w)$$

co jest równoważne następującej nierówności kończącej dowód monotoniczności ciągu $\{f(r_{i_j})\}$:

$$f(r_{i_a}(w)) \geq f(r_{i_b}(w))$$

□

Skonstruowanie pełnej relacji funkcyjnej niezależności dla skorzystania z powyższego twierdzenia może być trudne, dlatego łatwiej jest korzystać z warunków dostatecznych niezależności i traktować pary ruchów nie spełniające warunku dostatecznego jako ruchy potencjalnie wzajemnie funkcyjnie zależne.

W zadaniu przydziału prac warunkiem dostatecznym funkcyjnej niezależności ruchów lokalnych $\vec{r_i} = t_{i,i+1}$ i $\vec{r_j} = t_{j,j+1}$ jest warunek $|i-j| > 1$, bo dla $i, j = 1, \ldots n-1$

$$|i-j| > 1 \Longrightarrow \vec{r_i} \text{ jest funkcyjnie niezależny od } \vec{r_j},$$

co wynika z faktu, że transpozycje $t_{i,i+1}$ i $t_{j,j+1}$ z i oraz j takimi, że $|i-j| > 1$ mają wpływ na rozłączne zbiory pozycji permutacji ($\{i, i+1\} \cap \{j, j+1\} = \emptyset$ dla $|i-j| > 1$).

W zadaniu szukania gęstego podgrafu drogą długich, lecz prostych obliczeń można pokazać, że warunkiem dostatecznym funkcyjnej niezależności dwóch ruchów jest odpowiednia liczba krawędzi łączących wstawiane i usuwane podczas ruchów wierzchołki.

Dokładniej, ruch polegający na usunięciu wierzchołka v_1 i wstawieniu v_2 jest funkcyjnie niezależny od ruchu polegającego na usunięciu wierzchołka w_1 i wstawieniu w_2, jeśli suma liczb krawędzi łączących wierzchołki w_1 z v_1 i w_2 v_2 jest równa sumie liczb krawędzi łączących wierzchołki w_1 z v_2 i w_2 z v_1.

W badanym dalej algorytmie będziemy korzystać z jeszcze silniejszego warunku niezależności, tzn. będziemy zakładać, że ruch [usunięcie v_1, wstawienie v_2] jest potencjalnie zależny od ruchu [usunięcie w_1, wstawienie w_2], jeśli istnieje jakakolwiek krawędź między wierzchołkami ze zbioru $\{v_1, v_2\}$ a wierzchołkami ze zbioru $\{w_1, w_2\}$.

2.2. Ograniczenie liczby przeglądanych sąsiadów w wariancie najlepszego sąsiada

Załóżmy, że podczas szukania najlepszego sąsiada (najlepszego ruchu lokalnego) dla pierwszego wierzchołka utworzyliśmy hierarchię ruchów, tzn. strukturę złożoną ze wszystkich możliwych ruchów lokalnych dającą nam szybki dostęp do najlepszego z nich.

Najlepszy ruch, to taki, który daje największy przyrost wartości funkcji celu po jego wykonaniu. Oznaczmy przyrost wartości funkcji celu f w punkcie v na skutek ruchu r jako $\Delta_r f(v) \stackrel{def}{=} f(r(v)) - f(v)$.

Załóżmy, że dla naszej struktury istnieją mało kosztowne obliczeniowo – w porównaniu z obliczaniem wartości funkcji celu – operacje usuwania i wstawiania ruchów do hierarchii.

Wówczas, podczas szukania dominatora (najlepszego sąsiada) dla kolejnego wierzchołka na ścieżce dominatorów można skorzystać z informacji zgromadzonych w proponowanej strukturze danych.

Pomysł jest następujący. Jeśli dla wierzchołka v hierarchia miała przykładową postać $\Delta_{r_1} f(v) \geq \ldots \geq \Delta_{r_k} f(v)$, to po wykonaniu ruchu wskazanego przez r_1 (najlepszego ruchu) hierarchia pozostanie bez zmian z wyjątkiem pozycji dotyczących ruchów zależnych od ruchu wskazanego przez r_1, tzn. zgodnie z twierdzeniem $\Delta_{r_{i_1}} f(v) \geq \ldots \geq \Delta_{r_{i_m}} f(v)$, gdzie $\{r_{i_j}\}_{j=1}^{m}$, to ruchy funkcyjnie niezależne od r_1.

Zatem dla wszystkich ruchów zależnych od ruchu wskazanego przez r_1 wystarczy znaleźć nowe miejsce w tej częściowej hierarchii zależne od wartości funkcji celu, a będziemy mieć strukturę gotową do ponownego szybkiego wyboru najlepszego ruchu.

Załóżmy, że liczba różnych ruchów lokalnych przy jednorodnej funkcji sąsiedztwa wynosi k. Ponadto załóżmy, że liczba ruchów zależnych od jakiegokolwiek pojedynczego ruchu nie przekracza l. Przyjmijmy, że koszt obliczenia wartości funkcji celu wynosi $c(f)$. Wówczas stosując uporządkowane, zrównoważone struktury drzewiaste (np. drzewa czerwono-czarne lub 2-3 drzewa) zamiast obliczać wartość funkcji w k punktach ponosząc koszt $kc(f)$ wystarczy:

- odczytać i wykonać najlepszy ruch (koszt stały – $O(1)$, bo najlepszy ruch znajduje się w korzeniu struktury),
- usunąć ruchy zależne (koszt $O(l \log k)$),
- wyliczyć i wstawić nowe wartości ruchów zależnych (koszt $lc(f) + O(l \log k)$),

co daje łączny koszt $lc(f) + O(l \log k)$. Jest to zwykle mniej niż $kc(f)$, bo zazwyczaj l jest dużo mniejsze od k, zaś koszt obliczania wartości funkcji celu jest dużo większy od kosztu wykonywania operacji na strukturze drzewiastej.

Oczywiście, dodatkowym kosztem pokazanej metody w porównaniu z klasycznym poszukiwaniem lokalnym z wyborem najlepszego sąsiada jest koszt utworzenia początkowej hierarchii ruchów lokalnych, co dodatkowo wymaga posortowania ruchów. Innym dodatkowym kosztem może się okazać jednorazowe przygotowanie dla każdego ruchu lokalnego listy ruchów (potencjalnie) zależnych od niego w przypadku, gdy ruchów tych nie można szybko znaleźć. W zadaniu przydziału prac przygotowania takie są zbędne, w zadaniu szukania gęstego podgrafu wymaga to wstępnego przygotowania list ruchów potencjalnie zależnych od poszczególnych ruchów.

Poniżej prezentujemy zmodyfikowany algorytm LP demonstrujący wykorzystanie ruchów zależnych podczas procesu konstruowania ścieżki polepszającej.

ALGORYTM LPZ (LP wykorzystujący ruchy Zależne)

1. wybierz punkt startowy v {losowo lub według pewnej reguły};
2. utwórz hierarchię ruchów taką, że $\Delta_{r_1} f(v) \geq \Delta_{r_2} f(v) \geq \ldots \geq \Delta_{r_k} f(v)$;
3. **repeat**
4. wylicz punkt w - dominatora v z zależności: $w = r_1(v)$;
5. **if** $f(w) > f(v)$ **then**
6. zastąp punkt v punktem w;
7. usuń ruchy zależne od r_1 z hierarchii,
8. wstaw je ponownie z zachowaniem porządku;

   else
9. punkt v jest lokalnym maksimum f;
10. **until** punkt v będzie lokalnym względem sąsiedztwa maksimum f

Zauważyć można, że w hierarchii wystarczy pamiętać jedynie ruchy dające dodatni przyrost wartości funkcji celu, co dodatkowo zmniejszy koszt jej utrzymywania. Biorąc pod uwagę statystyczną zależność, że liczba lepszych sąsiadów maleje podczas posuwania się po ścieżce polepszającej, otrzymujemy dodatkowe, statystyczne przyspieszenie w końcowej fazie poszukiwania dominatorów.

Przedstawioną technikę przyspieszenia procesu konstruowania ścieżki polepszającej zastosowano w problemie szukania gęstego podgrafu, aby zbadać w praktyce, jaki efekt daje zmniejszenie liczby obliczeń wartości funkcji celu kosztem utrzymywania dodatkowych struktur danych. Ruchy lokalne zdefiniowano tak, jak wcześniej w przykładach.

Porównywano zachowanie algorytmu LP szukania gęstego podgrafu opartego na prostym szukaniu najlepszego sąsiada z algorytmem LPZ opartym na szukaniu z utrzymywaniem hierarchii ruchów. Wybrano dane dające reprezentatywne dla wielu prób wyniki. Zachowanie algorytmów badano dla grafów losowych z n wierzchołkami, z $\lfloor \binom{n}{2} \rho \rfloor^2$ losowo rozmieszczonymi krawędziami, z liczbą wierzchołków podgrafów równą $\lfloor n/2 \rfloor$, z progiem wymaganej liczby krawędzi mającym wartość najlepszego rozwiązania znajdowanego przez algorytm LP w czasie 1 minuty na komputerze z procesorem Pentium 133MHz.

W poniższej tabeli podano wartość liczonej w 100 próbach (po 10 uruchomień obu algorytmów dla 10 losowych grafów) średniej liczby obliczeń wartości funkcji f (L) dla

LP i LPZ, średniego czasu obliczeń (T) dla LP i LPZ oraz wartości ilorazów tych wielkości.

n	ρ	$L(LP)$	$L(LPZ)$	$L(LPZ)/L(LP)$	$T(LPZ)/T(LP)$	$T(LP)$	$T(LPZ)$
40	$\frac{1}{8}$	7398	4722	63.8%	73.3%	0.15s	0.11s
80	$\frac{1}{8}$	99796	56511	56.6%	60.3%	3.90s	2.35s
40	$\frac{1}{16}$	17385	7143	41.1%	47.2%	0.36s	0.17s
80	$\frac{1}{16}$	301051	101007	33.6%	36.0%	11.63s	4.19s

Zmniejszenie gęstości krawędzi zwiększa korzyści z zastosowania LPZ w porównaniu z LP, zwiększanie – zmniejsza. Przybliżoną gęstością, dla której zrównują się liczby obliczeń wartości funkcji, jest $\rho \approx \frac{1}{4}$, czasy działania algorytmów zrównują się dla $\rho \approx \frac{1}{5}$. Dla prób z $n = 40$ wartość odchylenia standardowego po grafach dla ilorazów wielkości nie przekraczała wartości 0.05, zaś dla prób z $n = 80$ nie przekraczała 0.01.

3. Podsumowanie

Prezentowana metoda pozwala zmniejszyć liczbę obliczeń wartości funkcji celu wymaganych do znalezienia lokalnego optimum znajdowanego algorytmem wielokrotnego przechodzenia do najlepszego sąsiada. Uzyskane przyspieszenie nie jest bezpośrednio zależne od rozwiązywanego problemu, lecz zależy głównie od struktury zdefiniowanego sąsiedztwa. W przypadku niektórych sąsiedztw liczba ruchów zależnych jest tak duża, że stosowanie tej metody powoduje wydłużenie czasu obliczeń o czas wykonania dodatkowych czynności. Nawet w takich przypadkach liczba obliczeń wartości funkcji nie jest większa niż w algorytmie podstawowym.

Dla wielu sąsiedztw definiowanych podczas rozwiązywania różnych problemów (problem komiwojażera, problem dostaw itp.) można uzyskać zmniejszenie liczby obliczeń wartości funkcji celu podczas szukania lokalnego optimum. W przypadku danych, dla których duża część par ruchów lokalnych jest funkcyjnie niezależna, otrzymamy w w efecie skrócenie czasu obliczeń.

Zmodyfikowane poprzez użycie prezentowanej metody poszukiwanie lokalne znajduje te same rozwiązania co standardowe wyszukiwanie lokalne z wyborem najlepszego sąsiada.

Część przytoczonych w pracy algorytmów i stosowanych pojęć opisano dokładniej i zilustrowano przykładami w pracy [1] .

LITERATURA

1. Aarts E., Lenstra J.K. (editors): Local Search in Combinatorial Optimization. John Wiley & Sons, 1997, pp. 1-18, 121-136, 215-310.

Recenzent: Dr hab.inż. Konrad Wala, prof. AGH

Abstract

Local search and tabu search are common used heuristics in solving of hard practical optimization problems. Presented method, in some cases, allow to achieve shorter time of searching desired solution or allow to find better solution in the same time. It may be applied to improve local search or tabu search based on finding the best neighbour.

It is introduced conception of functional independent moves and is proved the theorem, which yields improvement of local search and tabu search as its claim.

Results of applying this improvement for the practical problem of finding subgraph with a given number of vertices and with possible big number of edges are presented.