Polskie Towarzystwo Badań Operacyjnych i Systemowych
Instytut Badań Systemowych Polskiej Akademii Nauk
Wojskowa Akademia Techniczna

Redaktorzy:
Zbigniew Nahorski
Marian Chudy
Andrzej Straszak

OPTYMALIZACJA

Warszawa 1991

POLSKIE TOWARZYSTWO
BADAŃ OPERACYJNYCH I SYSTEMOWYCH

INSTYTUT BADAŃ SYSTEMOWYCH
POLSKIEJ AKADEMII NAUK

WOJSKOWA AKADEMIA TECHNICZNA

OPTYMALIZACJA

ZADANIA, METODY, ALGORYTMY

Redaktorzy

Zbigniew Nahorski, Marian Chudy, Andrzej Straszak

WARSZAWA 1991

OPTYMALIZACJA:
Zadania, metody, algorytmy
PTBOiS, Warszawa 1991.

METODA SZUKANIA O ZMIENNEJ GŁĘBOKOŚCI DLA PERMUTACYJNYCH ZAGADNIEŃ OPTYMALIZACJI

Piotr Kadłuczka , Konrad Wala
Instytut Automatyki AGH
al. Mickiewicza 30, 30-059 Kraków

Streszczenie: W pracy zaprezentowano aproksymacyjny algorytm optymalizacji przeznaczony do rozwiązywania permutacyjnych zagadnień optymalizacji. Algorytm zawiera trzy niezależne procedury, z których dwie realizują wieloiteracyjny proces poprawiania rozwiązania metodą poszukiwania o zmiennej głębokości, natomiast trzecia umożliwia wyjście procesu szukania z lokalnego optimum.

1. Wprowadzenie

NP-trudne zagadnienia optymalizacji kombinatorycznej dużego rozmiaru rozwiązuje się za pomocą algorytmów aproksymacyjnych. Jeżeli jakość rozwiązania otrzymanego przy pomocy reguły zachłannej nie jest zadawalająca, można wtedy zastosować dwuetapowy proces optymalizacji .

W pierwszym etapie obliczeń wyznaczone zostaje dopuszczalne rozwiązanie zagadnienia za pomocą wybranej metody inauguracyjnej, która charakteryzuje się tym, że podczas wykonania jednej iteracji ustalana jest jedna składowa rozwiązania. Dopuszczalne rozwiązania można otrzymać za pomocą reguły zachłannej lub innej procedury heurystycznej, a także na drodze losowania.

Etap drugi optymalizacji polega na zastosowaniu wieloiteracyjnego procesu poprawiania rozwiązania. Do tego celu stosowane są różne warianty metod optymalizacji lokalnej, algorytmy genetyczne oraz termodynamiczne.

W pracy zaprezentowano wieloiteracyjny algorytm poprawiania rozwiązania zagadnień optymalizacji kombinatorycznej, w których

rozwiązanie jest sformalizowane w postaci permutacji, tj. permutacyjnych zagadnień optymalizacji. Algorytm ten zawiera trzy niezależne procedury, z których dwie realizują iteracyjny proces poprawiania rozwiązania, natomiast trzecia procedura realizuje wyjście procesu szukania z lokalnego optimum. W iteracyjnych procedurach poprawiania rozwiązania zastosowano ideę poszukiwania o zmiennej głębokości, którą B.W.Kernighan i S.Lin zaproponowali po raz pierwszy w pracy [1], dla zagadnienia podziału grafu. W pracy [2] metodę poszukiwania o zmiennej głębokości zastosowano do rozwiązywania permutacyjnych zagadnień harmonogramowania.

NP-trudnymi permutacyjnymi zagadnieniami optymalizacji są takie, standardowe już zagadnienia, jak zagadnienie komiwojażera, kwadratowe zagadnienie przyporządkowania, liczne zagadnienia harmonogramowania procesów, zagadnienia równoważenia linii montażowych, itp.

2. Poszukiwanie o zmiennej głębokości

Niech $x=(x_1,\ldots,x_n)$, $x\in X$, będzie rozwiązaniem zagadnienia optymalizacji kombinatorycznej, X jest zbiorem wszystkich możliwych rozwiązań zagadnienia (przestrzeń poszukiwania rozwiązania) oraz S, $S\subset X$, jest zbiorem dopuszczalnych rozwiązań.

Istotą metody optymalizacji lokalnej jest próba poprawienia sukcesora x (najlepszego znanego rozwiązania zagadnienia) na drodze przeglądu zupełnego rozwiązań y z jego otoczenia N(x), $y \in N(x)$, gdzie $N(x) \subset X$ lub $N(x) \subset S$.

W metodzie poprawiania sukcesora na drodze przeszukiwania jego otoczenia natrafiamy na następującą sprzeczność. Jeżeli zdefiniujemy otoczenie w ten sposób, że liczba elementów zbioru N(x) nie jest duża, to proces poprawiania rozwiązania kończy się szybko i, zwykle, otrzymane rozwiązanie nie jest wystarczająco dobre - w "małym otoczeniu" trudno jest poprawić sukcesor. Jeżeli natomiast zbiór N(x) jest zbiorem licznym, to przegląd zupełny rozwiązań tego zbioru jest czasowo kosztownym. Jedną z prób rozwiązania tej sprzeczności jest interesująca metoda, którą B.W.Kernighan i S.Lin zaproponowali dla zagadnień podziału grafu

[1]. Przyjmując, że rozwiązaniem zagadnienia optymalizacji jest permutacja $x = (x_1, x_2, \ldots, x_n)$, tj. ciąg liczb całkowitych, który spełnia następujące warunki :

(i) $\forall\ i \in (1,2,\ldots,n) : x_i \in (1,2,\ldots,n)$

(ii) $\forall\ i \neq j : x_i \neq x_j$,

to rozwiązania z jej otoczenia generuje się zmieniając, w ściśle zdefiniowany sposób, położenie poszczególnych elementów w permutacji. Poniżej podano definicję otoczenia 'o zmiennej głębokości' dla permutacji, w każdym otoczeniu określono metodę 'szukania w głąb' i w ten sposób otrzymano dwie iteracyjne procedury, procedurę KL1 i KL2 (używana notacja : KLR, (R=1,2)), szukania o zmiennej głębokości, które są zasadniczymi składowymi prezentowanego niżej algorytmu aproksymującego APROXKW.

Poziomy głębokości szukania ponumerujemy indeksem l (l=1,2,...,L; L - maksymalna głębokość szukania), najlepsze rozwiązanie znalezione na głębokości l oznaczymy symbolem x^l; x^0 jest wtedy sukcesorem. Otoczenie pierwszego rodzaju rozwiązania x^{l-1} na głębokości l oznaczone jest symbolem $N1_l(x^{l-1})$, natomiast otoczenie drugiego rodzaju jest oznaczone $N2_l(x^{l-1})$; używana notacja $NR_l(x^{l-1})$, (R=1,2).

Rozwiązania y pierwszego otoczenia dla głębokości l = 1, $y \in N1_1(x^0)$ generowane są za pomocą zmian położenia elementów rozwiązania x^0 zdefiniowanych przez Regułę 1.

Reguła 1. Rozwiązanie y otrzymujemy z rozwiązania $x = (x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n)$ w ten sposób, że element x_i (i=1,2,...,n) jest przestawiany na miejsce elementu x_j, $j \neq i$, (j=1,2,...,n) oraz :

(i) jeżeli i<j to podciąg $x_{i+1}, x_{i+2}, \ldots, x_j$ jest przesuwany, w rozwiązaniu x, o jedną pozycję w lewo,

(ii) jeżeli i>j to podciąg $x_j, x_{j+1}, \ldots, x_{i-1}$ jest przesuwany o jedną pozycję w prawo.

Załóżmy, że dokonano przeglądu wszystkich rozwiązań otoczenia $N1_1(x^0)$ oraz niech przestawienie elementu x_i na miejsce elementu x_j (j≠i) w rozwiązaniu x^0 daje rozwiązanie x^1, które jest najlepsze w zbiorze $N1_1(x^0) \cap S$; $f(x^1) = \min\{f(y): y \in N1_1(x^0) \cap S\}$,

$f(y)$ - funkcja celu. Dalsze poszukiwanie w głąb lepszych rozwiązań następuje już w otoczeniu $Ni_2(x^1)$, które otrzymujemy w następujący sposób. W rozwiązaniu x^1 ustalane jest położenie elementu x_i (rozwiązania x^0) wśród jego bezpośrednich sąsiadów z lewej i prawej strony. Jeżeli $i<j$ ($j=n$) to położenie elementu x_i w rozwiązaniu x^1 jest następujące :

$$x^1 = (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_{j-1},\underline{x_j,x_i,x_{j+1}},x_{j+2},\ldots,x_n)$$

a w przypadku $i>j$ ($j=1$) odpowiednio :

$$x^1 = (x_1,\ldots,x_{j-1},\underline{x_j,x_i,x_{j+1}},x_{j+2},\ldots,x_{i-1},x_{i+1},\ldots,x_n)$$

W ten sposób w rozwiązaniu x^1 istnieje trójka $[x_j,x_i,x_{j+1}]$, w której kolejność jest ustalona i przy dalszym poszukiwaniu 'dobrych' rozwiązań, zarówno w otoczeniu $Ni_2(x^1)$, jak i w następnych zbiorach $Ni_1(x^{l-1})$ ($l=3,4,\ldots,L$), nie może być zmieniona. W przypadku, gdy element x_i wstawiany jest na miejsce elementu x_1 lub x_n to jego położenie jest ustalane tylko w stosunku do jednego sąsiada tj. odpowiednio prawego lub lewego.

Podciągi, w których kolejność nie jest zmieniana w czasie generowania nowych rozwiązań będziemy nazywać podciągami ustalonymi, używając przy tym do notacji symbolu [.....].

Podczas generowania rozwiązań z otoczenia $Ni_2(x^1)$ element x_i i jego bezpośredni sąsiedzi (tj. ustalony w rozwiązaniu x^1 podciąg $[x_j,x_i,x_{j+1}]$) traktowane są jako jedna całość. Oznacza to, że przy kolejnych zmianach położenia elementów permutacji, zgodnie z zasadami określonymi przez Regułę 1, przestawieniu lub przesunięciu ulega cały podciąg ustalony.

Niech przestawienie elementu x_k w permutacji x^1 daje najlepsze rozwiązanie x^2: $f(x^2) = \min\{f(y): y \in Ni_2(x^{l-1}) \cap S\}$. Jeżeli sąsiadem przestawionego elementu x_k jest ustalony podciąg to zostaje on w całości składnikiem nowego ustalonego podciągu, np. w przypadku $x^2 = (x_1,\ldots,\underline{x_r,x_k,[x_j,x_i,x_{j+1}]},x_{r+1},\ldots,x_n)$ otrzymujemy następujący podciąg ustalony $[x_r,x_k,x_j,x_i,x_{j+1}]$.

Powyższa zasada ustalania podciągów w rozwiązaniach x^l, $l=3,4,\ldots,L$, jest realizowana przy kontynuowaniu szukania w głąb. Każde otoczenie $N_1(x^{l-1})$ rozwiązań x^{l-1} jest wyznaczone za pomocą zmian położenia elementów rozwiązań x^{l-1} zdefiniowanych przez

Regułę 1, przy czym ustalone podciągi [.....] rozwiązań x^{l-1} ulegają w całości przestawieniu lub przesunięciu.

Rozwiązania y drugiego rodzaju otoczenia generowane są za pomocą zmian położenia elementów permutacji zdefiniowanych przez Regułę 2.

Reguła 2. Rozwiązanie y otrzymujemy z rozwiązania $x=(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n)$ w ten sposób, że element x_i $(i=1,2,\ldots,n)$ jest przestawiany na miejsce elementu x_j, $j\neq i$, $(j=1,2,\ldots,n)$ oraz element x_j na miejsce elementu x_i; $y=(x_1,\ldots,x_j,\ldots,x_i,\ldots,x_n)$.

Rozwiązania $y \in N2_1(x^0)$ generowane są za pomocą zmian położenia elementów rozwiązania x^0 określonych przez Regułę 2. Niech przestawienie elementów x_i i x_j $(i\neq j)$ daje rozwiązanie x^1, które jest najlepsze w zbiorze $N2_1(x^0) \cap S$; $f(x^1)= \min \{f(y): y \in N2_1(x^0) \cap S\}$. Ustalamy dalej pozycje 'i', 'j' w rozwiązaniu $x^1=(x_1,\ldots,\underline{x_j},\ldots,\underline{x_i},\ldots,x_n)$. Oznacza to, że w czasie generowania rozwiązań z otoczeń $N2_1(x^{l-1})$, $(l=2,3,\ldots,L)$ za pomocą przestawiania elementów rozwiązań x^{l-1} określonych przez Regułę 2, elementy znajdujące się na ustalonych pozycjach nie są przestawiane. Jeżeli rozwiązanie $x^2 \in N2_2(x^1) \cap S$ otrzymano z rozwiązania x^1 przestawiając elementy x_k, x_l, $k \neq l$ to w rozwiązaniu x^2 ustalane są kolejne dwie pozycje, mianowicie pozycje 'k' i 'l'. Powyższa zasada ustalania pozycji rozwiązań x^l $(l=2,3,\ldots,L)$ jest realizowana przy kontynuowaniu szukania w głąb.

Dla tak zdefiniowanych otoczeń $NR_1(x)$, $(R=1,2)$, procedury poszukiwania o zmiennej głębokości KLR, $(R=1,2)$, mają postać :

'Procedura KLR (R = 1 lub 2)

Krok 1. Ustalamy głębokość szukania L $(L=1,2,\ldots)$ oraz podstawiamy $l := 1$, $x^{l-1} = x^0 := x$

Krok 2. Rozwiązujemy zagadnienie (metodą przeglądu zupełnego otoczenia):
$f(x^1) = \min \{f(y): y \in NR_1(x^{l-1}) \cap S\}$
oraz określamy otoczenie $NR_{l+1}(x^1)$

Krok 3. Powtarzamy krok 2 dla $l=2,3,\ldots,L$. Wybieramy takie $k \in \{1,2,\ldots,L\}$, dla którego funkcja celu $f(x^k)$ jest minimalna

Krok 4. Jeżeli $f(x^k) < f(x^0)$, to podstawiamy $x^0 := x^k$, $l := 1$ i powtarzamy obliczenia od kroku 2, w przeciwnym przypadku procedura KLR kończy działanie.

gdzie : $f(y)$ - funkcja celu określona dla $y \in S$,
x - najlepsze znane rozwiązanie (początkowy sukcesor).

Uwagi :

(1) Znalezione w kroku 2 rozwiązanie x^l nie musi być lepsze od rozwiązania x^{l-1}.

(2) Jeżeli L=1 to procedury LK1 i LK2 są równoważne standardowym procedurom optymalizacji lokalnej.

(3) Dla rozwiązań określonych w postaci n elementowej permutacji, $x=(x_1,\ldots,x_n)$, liczba rozwiązań w otoczeniu $N1_l(x)$ zawiera się w przedziale :

$$|N1_l(x)| \in \langle [n-2(l-1)][n-2(l-1)-1] \ , \ [n-(l-1)][n-(l-1)-1] \rangle =$$
$$\langle (n-2l+2)(n-2l+1) \ , \ (n-l+1)(n-l) \rangle,$$

natomiast dla otoczenia $N2_l(x)$ wynosi :

$$|N2_l(x)| = \tfrac{1}{2}[n-2(l-1)][n-1-2(l-1)] = \tfrac{1}{2}(n-2l-2)(n+1-2l).$$

Stąd wspólne ograniczenie dla KL1 i KL2 na maksymalną głębokość szukania ma postać : $L \le \lfloor \frac{1}{2}n \rfloor$.

Algorytm aproksymujący APROXKW poprawiania rozwiązań stosuje na przemian procedury KL1 i KL2 tak długo, dopóki nie jest osiągnięte optimum lokalne procedur, tj. do chwili, gdy kolejne użycie obu procedur nie prowadzi do poprawy rozwiązania. Wtedy stosowana jest procedura RAN, która losuje rozwiązanie "bliskie" do otrzymanego lokalnie optymalnego rozwiązania. Tak uzyskane rozwiązanie algorytm APROXKW ponownie próbuje poprawić za pomocą procedur KL1 i KL2.

3. Zadania testowe

W pierwszej kolejności funkcjonowania algorytmu sprawdzono na znanych zadaniach testowych kwadratowego zagadnienia przyporządkowania (QAP). Skorzystano tu z zadań zamieszczonych w pracy [4]. Minimalizowana funkcja celu ma postać :

$$\sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{x_i x_k}$$

gdzie :

$[a_{ik}]_{nxn}$ – macierz odległości pomiędzy pozycjami rozmieszczenia przedsiębiorstw,

$[b_{jl}]_{nxn}$ – macierz wzajemnej zależności lub przepływu pomiędzy przedsiębiorstwami j i l.

$S = X$ – w zagadnieniu QAS zwykle dopuszczalne są wszystkie permutacje.

Otrzymano następujące wyniki dla danych testowych zamieszczonych w [4] :

(a) zadanie o wymiarze n=15 – najlepsze z dotychczas znalezionych rozwiązań f=1160, uzyskano lepsze wyniki dla rozwiązań :

f=1150

Nr pozycji :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1) Nr elementu :	10	15	6	5	12	4	3	14	7	11	1	2	13	8	9
2) Nr elementu :	1	2	13	8	9	4	3	14	7	11	10	15	6	5	12
3) Nr elementu :	9	8	13	2	1	11	7	14	3	4	12	5	6	15	10
4) Nr elementu :	12	5	6	15	10	11	7	14	3	4	9	8	13	2	1

f=1152

5) Nr elementu :	1	2	4	14	6	9	13	3	5	15	11	8	7	12	10

f=1158

6) Nr elementu :	1	2	13	8	9	4	3	14	7	11	10	6	15	5	12

(b) zadanie o wymiarze n=20 – publikowane rozwiązanie f=2574, lepsze wyniki uzyskano dla następujących rozwiązań :

f=2570

Nr pozycji :	1	2	3	4	5	6	7	8	9	10
1) Nr elementu :	9	3	10	14	18	16	11	12	2	4

	Nr pozycji :	11	12	13	14	15	16	17	18	19	20
	Nr elementu :	13	8	20	15	19	6	1	7	5	17
2)	Nr elementu :	6	1	7	5	17	13	8	20	15	19
		16	11	12	2	4	9	3	10	14	18

(c) zadanie o wymiarze n=30 - publikowane rozwiązanie f=6164, lepsze wyniki uzyskano dla następujących rozwiązań :

f=6142

	Nr pozycji :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1)	Nr elementu :	12	6	25	13	5	28	24	26	10	9	2	21	1	22	7
	Nr pozycji :	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	Nr elementu :	19	29	20	17	18	8	16	3	14	15	23	11	30	27	4

f=6153

2)	Nr elementu :	5	2	29	3	4	14	28	21	16	19	30	20	25	9	7
		8	11	27	6	13	10	22	23	18	12	24	26	1	17	15

Są to najlepsze, jak dotychczas, rozwiązania wymienionych zadań testowych. Wyniki obliczeń związane z zastosowaniem algorytmu APROXKW do innych zagadnień są zamieszczone w pracy [3].

Literatura

[1] Kernighan B.W., Lin S.: An Efficient Heuristic Procedure for Partitioning Graphs. BSTJ, No.2, 1970,pp.291-307.

[2] Wala K.: Metody lokalnej optymalizacji w zagadnieniu harmonogramowania. Zeszyty Nauk. Politechniki Śląskiej, Automatyka z.93, Gliwice 1988, pp.169-177.

[3] Kadłuczka P., Wala K.: Wyniki testowania metody szukania o zmiennej głębokości na przykładach permutacyjnych zagadnień optymalizacji. Materiały IX Ogólnopolskiego Sympozjum Zastosowań Teorii Systemów, Krościenko '91 (w druku).

[4] Nugent Ch.E., Vollmann T.E., Ruml J.: An Experimental Comparison of Techniques for the Assignment of Facilities to Locations, Operation Research, Vol.16, 1968, pp.150-173.

ISBN 83-900412-1-9.