ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1984
Seria: AUTOMATYKA z. 74 Nr kol. 810

Leon Słomiński

Instytut Badań Systemowych PAN
Warszawa

ALGORYTMY ROZWIĄZUJĄCE MINIMAKSOWE ZAGADNIENIE PRZYDZIAŁU I ICH KOMPUTEROWE PORÓWNYWANIE

Streszczenie. Przedstawiono schemat, nazwany schematem progowym, który pozwala opisywać, w sposób jednolity i wygodny z punktu widzenia porównywania, znane algorytmy rozwiązujące minimaksowe zagadnienie przydziału. Podano przykłady zapisu algorytmów ujętych w schemat progowy. Omówiono wyniki komputerowych porównań algorytmów i wskazano na pewne niedoskonałości i sprzeczności uzyskiwanych tą drogą ocen.

Przedstawimy schemat nazwany schematem progowym, który pozwala w sposób jednolity i wygodny z punktu widzenia porównywania opisać znane algorytmy rozwiązujące minimaksowe zagadnienie przydziału. Podamy przykłady zapisu różnych algorytmów, ujętego w schemat progowy. Omówimy wyniki komputerowych porównań algorytmów i wskażemy na pewne sprzeczności w uzyskiwanych tą drogą ocenach.

Minimaksowe zagadnienie przydziału rozpatrujemy w następującej postaci:

$$\min \left\{ \max_{i,j/x_{ij}>0} \left\{ c_{ij}\, x_{ij} \right\} \right\} \qquad /ZPM/$$

przy ograniczeniach

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j=1,\ldots,n, \qquad \sum_{j=1}^{n} x_{ij} = 1, \quad i=1,\ldots,n$$

$$x_{ij} = 0 \quad \text{lub} \quad 1, \quad i,j=1,\ldots,n ,$$

gdzie: $c_{ij} \geq 0$, a minimum jest szukane po wszystkich przydziałach, tzn. po wszystkich macierzach $[x_{ij}]_{n \times n}$, które spełniają nałożone ograniczenia.

Minimaksowe zagadnienie przydziału jest klasycznym modelem w badaniach operacyjnych, a jego zastosowania praktyczne zarówno w postaci standardowej jak i w wersjach zmodyfikowanych, są omówione w wielu pozycjach literatury.

Minimaksową postać zagadnienia przyjmujemy dla ustalenia uwagi, zaś wszystkie stwierdzenia i wnioski, które przytaczamy dalej, można bez trudu przystosować do maksiminowej postaci zagadnienia.

1. Schemat progowy

Opis znanych algorytmów dla rozwiązania /MZP/ można sprowadzić do jednego prostego i uniwersalnego schematu, nazwanego przez nas schematem progowym. Schemat ten zapiszemy w postaci czterech kroków.

1. Wybór progu. Jeżeli zbiór elementów progowych /najczęściej tworzą go elementy macierzy C/ został wyczerpany to przejdź do kroku 4, w przeciwnym przypadku, kierując się regułą wyboru, wybierz element T, który posłuży jako próg.

2. Transformacja progowa. Podziel elementy macierzy C na dwa podzbiory: $C_d = \{c_{ij}/c_{ij} \leq T\}$ i $C_g = \{c_{ij}/c_{ij} > T\}$.

3. Test dopuszczalności. Wariant A: Czy macierz C_T utworzona z elementów dopuszczalnych macierzy C /zależnie od metody wyboru progu mogą to być elementy podzbioru C_d lub C_g/ ma przydział ? Wróć do kroku 1 /odpowiedź jest wykorzystana przy wyborze progu/.

Wariant B: Czy k pierwszych wierszy macierzy C_T /utworzonej jak wyżej/ dopuszcza k-przydział ? Jeżeli tak, to k:= k+1, powtórz test. W przeciwnym razie wróć do 1.

4. Koniec obliczeń. Otrzymano rozwiązanie optymalne. Wartość ostatniego progu jest wartością optymalną funkcji celu, a dowolny przydział dopuszczalny /dla progu optymalnego/ jest przydziałem optymalnym.

Z założeń /MZP/, według których macierz C jest macierzą pełną wynika, że rozwiązanie zagadnienia istnieje. Skończoność liczby iteracji schematu progowego jest zapewniona, gdyż macierz C jest skończona a każdy jej element może co najwyżej raz służyć jako próg /ogólna zasada obowiązująca dla reguł wyboru progu/.

Wyróżnimy trzy reguły wyboru progu:

- wybór monotoniczny: a/ z wartościami rosnącymi /$T_{k-1} < T_k$/, zwykle /aczkolwiek nie wyłącznie/ stosowany w przypadku kryterium minimaksowego; b/ z wartościami malejącymi /$T_{k-1} > T_k$/, stosowany zwykle /aczkolwiek nie wyłącznie/ w przypadku kryterium maksiminowego;

- wybór przemienny /polegający na kolejnym zwiększaniu i zmniejszaniu progu/: a/ dychotomiczny, który polega na sukcesywnym połowieniu aktualnego podzbioru elementów macierzy C i wyborze elementu środkowego /uwzględniając parzystość lub nieparzystość liczności podzbioru, a także możliwość występowania elementów identycznych/, b/ nieregularny, w którym próg jest wybierany przez niepołówkowy podział podzbioru /[11]/, c/ mieszany - monotoniczno-przemienny - który może się okazać przydatny dla zagadnień o bardzo dużych rozmiarach /np. $n \geq 1000$/.

Analiza nakładów obliczeń pokazuje /[11]/, że liczba progów jest $O(n)$ dla wyboru monotonicznego, $O(\log n)$ - dla wyboru dychotomicznego i leży między tymi wielkościami dla wyboru mieszanego.

Testy dopuszczalności będziemy formułować na dwóch obiektach związanych z macierzą C: macierzy C_T i digrafie dwudzielnym G_T. W macierzy C_T

elementy z C_d traktujemy jako jedynki i uważamy je za elementy dopuszczalne, natomiast elementy z C_ε są elementami niedopuszczalnymi i nadajemy im wartość zero.

Digraf dwudzielny $G_T(V_1,V_2;A)$ utworzony jest z podzbioru wierzchołków V_1 odpowiadających numerom wierszy macierzy C; podzbioru wierzchołków V_2 odpowiadających numerom kolumn /$|V_1| = |V_2| = n$/, oraz ze zbioru łuków, z których każdy łączy wierzchołek z V_1 z wierzchołkiem należącym do V_2, przy czym $(i,j) \in A$ wtedy i tylko wtedy, gdy w macierzy C element $c_{ij} = 1$, /$i \in V_1$, $j \in V_2$/.

Digraf G_T będziemy także rozszerzali do sieci S_T, uzupełniając G_T o wierzchołek-źródło s i wierzchołek-ujście t, oraz o łuki prowadzące z s do każdego $i \in V_1$ i o łuki z każdego $j \in V_2$ do t.

Użyty będzie również graf skierowany $G'_T(V,A')$, w którym zbiór V tworzą wierzchołki ponumerowane od 1 do n. Z wierzchołka i do wierzchołka j prowadzi łuk (i,j) wtedy i tylko wtedy, gdy $c_{ij} = 1$ /w macierzy C_T/ przy czym pętle, które odpowiadają $c_{ii} = 1$ są w grafie pomijane.

TESTY DOPUSZCZALNOŚCI NA MACIERZY C_T

1. Sprowadzić macierz C_T do postaci, w której na jednej z głównych przekątnych są same jedynki /znaleźć przydział/. Jeżeli jest to niemożliwe, to należy zmienić próg. Zadanie to można rozwiązać przy nakładzie obliczeń $O(n^{5/2})$.

1a. Podmacierz prostokątną, złożoną z k wierszy i n kolumn sprowadzić do postaci z samymi jedynkami na mniejszej diagonali /znaleźć k-przydział/. Jeżeli jest to niemożliwe, to należy zmienić próg. W przeciwnym razie test jest powtarzany dla podmacierzy o k+1 wierszach. Powiększenie k-przydziału do k+1-przydziału wymaga $O(n^2)$ obliczeń, a znalezienia n-przydziału można dokonać nakładem $O(n^3)$.

TESTY DOPUSZCZALNOŚCI NA DIGRAFACH G_T i G'_T.

1. W digrafie G_T znaleźć skojarzenie o mocy n. Jeżeli skojarzenie maksymalne jest mniejsze od n to należy zmienić próg. Znane algorytmy pozwalają rozwiązać ten problem nakładem obliczeń $O(n^{5/2})$.

1a. Zakładamy, że znamy pewne skojarzenie o mocy n /wartość progu T jest w tym przypadku oszacowaniem górnym dla szukanego minimum/. Należy znaleźć cykl elementarny o własnościach: startujemy od wierzchołka, który jest początkiem łuku o wadze T /w macierzy C/, każdy następny łuk ma wagę mniejszą od T i musi spełniać warunek przemienności, co oznacza, że jeżeli dany łuk należy do skojarzenia to sąsiadujące z nim /w cyklu/ łuki nie mogą do skojarzenia należeć. Istnienie takiego cyklu pozwala otrzymać nowy przydział, w którym rozważany łuk o wadze T nie wystąpi. Proces kontynuujemy eliminując wszystkie łuki o wadze T z przydziału, po czym powtarzamy test dla przydziału o mniejszej wartości T.

Znalezienie cyklu elementarnego w G_T kosztuje $O(n^2)$ obliczeń i podobny nakład, zwykle o mniejszych stałych jest potrzebny dla znalezienia cyklu elementarnego o najmniejszej liczbie łuków.

2. W digrafie G'_T należy znaleźć elementarny cykl skierowany, zaczynając od dowolnego wierzchołka, dla którego $c_{ii}=0$. Znalezienie takiego cyklu pozwala zwiększyć moc aktualnego k-przydziału przynajmniej o jedność. Jeżeli tak postępując /dla każdego $c_{ii}=0$/, nie otrzymamy n-przydziału, to należy zmienić próg, a jeżeli przydział otrzymamy, to próg jest zmieniany tylko wtedy, gdy zbiór progów nie został wyczerpany.

Nakład obliczeń na znalezienie jednego cyklu, jak również nakład niezbędny dla znalezienia cyklu najkrótszego, jest $O(n^2)$.

TEST DOPUSZCZALNY NA SIECI S_T

1. W sieci S_T znaleźć przepływ maksymalny ze źródła s do ujścia t. Jeżeli wartość tego przepływu wynosi n, to albo mamy rozwiązanie optymalne /wyczerpany zbiór progów/, albo należy zmienić próg. Jeżeli wartość przepływu maksymalnego jest mniejsza od n, to próg musi być zmieniony. Koszt znalezienia przepływu maksymalnego w zero-jedynkowej sieci S_T jest $O(n^{5/2})$.

Ze względu na realizację komputerową, wyróżnienie zadań testowych na macierzach i grafach /sieciach/ ma w zasadzie charakter metodologiczny. Ekonomiczne struktury danych, do zapisu i operowania wielkościami występującymi w zadaniach, z których korzysta się w komputerowych implementacjach algorytmów, zacierają różnice związane z językiem, w którym te zadania są zapisane.

2. Charakterystyka ważniejszych algorytmów

1. ALGORYTM GROSSA /G/, 1959 r, [9]. Wartość funkcji celu dowolnego przydziału początkowego staje się oszacowaniem górnym dla wartości optymalnej. Próg jest zmieniany monotonicznie w porządku malejącym. Zadaniem testowym jest zadanie 1a na digrafie G_T /lub, co jest zupełnie równoważne, na macierzy C_T/, tzn. szukany jest cykl elementarny. Łączny nakład obliczeń dla algorytmu G jest $O(n^4)$.

2. ALGORYTM FORDA-FULKERSONA /FF/, 1962 r, [5]. Algorytm ten nazywany jest także, niezbyt ściśle, algorytmem węgierskim i różni się od algorytmu G jedynie zastosowaniem zmodyfikowanej procedury etykietowania /oryginalna metoda wprowadzona przez Forda i Fulkersona dla znajdowania przepływu maksymalnego w sieciach/ do znajdowania cyklu elementarnego.

3. ALGORYTM PAPE I SHÖNA /PS/, 1970 r, [10]. Algorytm ten jest identyczny, pomijając mniej istotne szczegóły, z algorytmem Garfinkela, który charakteryzujemy pod pozycją 4.

4. ALGORYTM GARFINKELA /GARF/, 1971 r. [7]. Algorytm ten ma naturalną konstrukcję progową. Wartością początkową progu jest największy element spośród najmniejszych w poszczególnych wierszach i kolumnach. Próg jest zmieniany monotonicznie, w porządku rosnącym. Zadaniem testowym jest zagadnienie przepływu maksymalnego w zero-jedynkowej sieci S_T. W najprost-

szej wersji algorytmu nakład obliczeń dla przypadku najgorszego jest $O(n^{9,5})$. Ekonomiczniejszą wersję tego algorytmu, której nakład jest $O(n^3)$ podał Gordon /patrz: pozycja 5/.

5. ALGORYTM GORDONA /GORD/, 1976 r. [8]. Próg jest zmieniany według reguły monotonicznej, w porządku malejącym /sieć początkowa nie zawiera w ogóle łuków, ich liczba rośnie w miarę malenia progu/. Dla ułatwienia wyboru progu elementy macierzy C są jednorazowo uporządkowane w ciąg niemalejący /koszt: $O(n^2 \log n)$/. Zmniejszenie nakładu łącznego do wielkości $O(n^3)$, uzyskano przez odpowiednie wykorzystanie w kolejnej iteracji, informacji o negatywnym wyniku testu na iteracji poprzedzającej.

6. ALGORYTM SŁOMIŃSKIEGO /SIGMIN/, 1976 r. [11,12]. Próg jest wybierany tak jak w algorytmie GARF, z tym że zaproponowana implementacja tej reguły gwarantuje co najwyżej n zmian progu dla całego algorytmu. Test dopuszczalności polega na sprawdzeniu macierzy C_T do postaci z samymi jedynkami na głównej przekątnej. Realizowano go jako zadanie 2 na digrafie G'_T. W tej wersji koszt algorytmu SIGMIN jest $O(n^4)$. Zero-jedynkowa postać kolumn i wierszy macierzy C_T umożliwia, przy wykorzystaniu zapisu informacji na bitach słowa maszynowego, uzyskanie nakładu obliczeń $O(n^3)$ lub $O(n^2 \log n)$ - w przypadku dychotomicznej reguły wyboru progu.

7. ALGORYTM DERIGSA I ZIMMERMANNA /BAP1/, 1977 r. [4]. Próg rośnie w kolejnych iteracjach algorytmu. Próg początkowy jest szacowany podobnie jak w SIGMIN i w GARF. Test dopuszczalności polega na szukaniu k-przydziału /k=1,...,n/ w prostokątnej podmacierzy macierzy C_T, złożonej z k pierwszych wierszy. Znalezienie k-przydziału przy znajomości /k-1/-przydziału, lub stwierdzenie konieczności zmiany progu, kosztuje $O(n^2)$ operacji. Próg początkowy T_p jest wykorzystywany również do wyznaczania częściowego przydziału początkowego /wśród elementów $c_{ij} \leq T_p$/ obejmującego możliwie najwięcej wierszy /koszt $O(n^2)$/. Łączny nakład obliczeń dla algorytmu BAP1 jest $O(n^3)$.

8. ALGORYTM FIRKE I SMITHA /FS/, 1979 r. [6]. Progiem jest, każdorazowo pewne oszacowanie dolne z szukanej wartości optymalnej. Macierz C jest przekształcona następująco: $c'_{ij} = \lfloor c_{ij} / z+1 \rfloor$, gdzie $\lfloor x \rfloor$ oznacza największą liczbę całkowitą nieprzekraczającą x.

Test dopuszczalności polega na szukaniu przydziału o wartości optymalnej równej zeru /jeżeli $c_{ij}/(z+1) < 1$, to $c'_{ij} = 0$ w macierzy C_T, $T=z$/. Jeżeli przydział taki istnieje, to jest on przydziałem optymalnym, w przeciwnym przypadku wyznaczany jest nowy próg: $z' = \min_{i,j} \{ c_{ij} | (w_i+w_j) > 0 \} > z$, gdzie: u_i oraz w_j są zmiennymi dualnymi związanymi z otrzymanym przydziałem y_{ij}, które spełniają nierówności: $(u_i+w_j) \leq c'_{ij}$, $i,j=1,\ldots,n$ oraz $y_{ij} > 0 \Rightarrow (u_i+w_j) = c'_{ij}$.

9. ALGORYTM CARPANETO I TOTHA /BASS/, 1981 r. [2]. Algorytm ten jest nieznaczną modyfikacją algorytmu BAP1. Różnica polega na prostszym sposo-

bie wyznaczania początkowego przydziału częściowego i na niektórych drobnych zmianach w implementacji zadania testowego. Nakład obliczeń algorytmu jest $O(n^3)$.

3. O komputerowych porównywaniach algorytmów dla /MZP/

Algorytmy dla rozwiązania minimaksowego zagadnienia przydziału były porównywane między sobą. W [7] podano wyniki porównawcze dla algorytmów G i GARF, w [4] - dla GARF i BAP1, w [2] - dla BAP1 i BASS, a w [3] - dla GARF, FS i BAP1. Oddzielnie zajmiemy się wynikami porównań algorytmów BAP1 i SIGMIN.

Typowy eksperyment porównawczy polega na zestawieniu czasów obliczeń /rzadziej liczby iteracji/ dla pewnej liczby przykładów /od kilkudziesięciu do kilkuset/ o różnych wartościach parametrów n i b, gdzie $b=\max_{i,j}\{c_{ij}\}$. Przykłady są tworzone przez zapełnianie pozycji macierzy C liczbami losowymi, z przedziału [1,b], o rozkładzie równomiernym.

Na podstawie eksperymentów wykonanych przy takich ogólnych założeniach, autorzy cytowanych wyżej prac dochodzą do pewnych powtarzających się wniosków. Zadania generowane losowo przy założeniu równomierności rozkładu są zadaniami "łatwymi", co uwidacznia się w tym, że rzeczywisty nakład obliczeń rośnie raczej proporcjonalnie do n^2, niezależnie od teoretycznej oceny najgorszego przypadku mającej rząd $n^3 \div n^5$ /zależnie od algorytmu /. Obserwuje się umiarkowany wzrost czasu obliczeń w funkcji b /zależność ta jest nieco silniejsza dla algorytmu GARF /PS/ i słabsza dla - BAP1 i BASS/. Wszystkie implementacje potwierdzają celowość korzystania z oszacowania dolnego dla wartości optymalnej i uzyskanie go wymaga niedużego nakładu obliczeń. Większą wrażliwość na jakość tego oszacowania wykazują algorytmy GARF /PS/, FS i SIGMIN, zaś mniejszą - BAP1 i BASS. Algorytmy: G,BAP1, BASS wymagają zwykle znacznie więcej iteracji niż pozostałe algorytmy, niemniej czasy trwania iteracji są raczej nieporównywalne. Znajomość częściowego rozwiązania początkowego /monotonicznie rosnący próg/ - k-przydziału skraca czas podstawowych obliczeń, z tym że dyskusyjny może być nakład obliczeń, który jesteśmy gotowi ponieść dla uzyskania tego rozwiązania. Z naszej oceny wynika na przykład, że cały zysk czasowy algorytmu BASS w stosunku do BAP1, powstaje wskutek zrezygnowania z porządkowania wierszy przy ustalaniu początkowego k-przydziału.

W kilku pracach / np. [2], [4], [7], [10] / pojawia się konkluzja końcowa mówiąca o tym, że przeprowadzony test świadczy o bezwzględnej przewadze danego algorytmu nad innym algorytmem. Jest to teza wątpliwa, szczególnie w świetle wyników porównań przedstawionych w pracach [1], [3], [14] oraz wyników analizy dokonanej w [13]. W tabelach 1 i 2 podajemy najważniejsze wyniki porównań algorytmów SIGMIN i BAP1. Rezultaty tabeli 1 uzyskano w następujących warunkach: komputer IBM 370/145, system wieloprocesо-

rowy, język FORTRAN IV. Liczby c_{ij} otrzymano przez transformację do przedziału [0,b] liczb generowanych losowo, niezależnie, z jednakowym prawdopodobieństwem z przedziału [0,67101323]. Czas obliczeń jest w każdym przypadku średnią arytmetyczną otrzymaną z trzech zadań.

Tabela 1

n \ b	100		1000		10000	
	Czas w sekundach					
	SIGMIN	BAP1	SIGMIN	BAP1	SIGMIN	BAP1
10	0.034	0.019	0.021	0.013	0.024	0.016
20	0.074	0.053	0.074	0.054	0.069	0.050
50	0.402	0.437	0.431	0.512	0.398	0.401
80	0.990	1.234	1.984	1.661	3.107	1.630
100	1.566	1.795	1.535	2.372	1.552	1.695
150	3.891	6.555	5.419	7.368	3.572	3.891
200	5.778	9.064	14.139	9.432	5.969	16.771
250	9.305	18.818	9.407	14.629	9.327	16.139

Analizując wyniki tabeli 1 zauważamy, że:
- Dla n=10 i n=20 czasy uzyskane przez SIGMIN są gorsze od czasów dla BAP1.
- Dla n=50-250, na 18 wyników 3 razy algorytm SIGMIN miał czas gorszy od BAP1 /przed uśrednieniem wyników, na 72 testy SIGMIN przeważał w 41 przypadkach, z tym że dla $n \geq 50$, na 55 testów , SIGMIN przeważał 45 razy/.
- Nie obserwuje się wyraźnego wzrostu czasu obliczeń ze wzrostem b; dla $n \geq 50$ czas ten nawet maleje /fakt ten jest wyjaśniony w [13]/.

Rezultaty tabeli 2 uzyskano w następujących warunkach: komputer CDC CYBER 76, system jednoprocesorowy, język FORTRAN IV. Liczby c_{ij} generowano w przedziale $[1,2^{31}-1]$ i następnie transformowano je do przedziału [1,b]. Tabela podaje w kolejności: czas najlepszy, średni i najgorszy, otrzymany w każdym przypadku z uśrednienia 25 wyników.

Wnioski autora wyników są następujące. Jeżeli początkowe oszacowanie dolne jest równe wartości optymalnej /oszacowania te otrzymuje się identycznie dla obydwóch algorytmów i w 90 % przypadków było ono wartością optymalną/ to SIGMIN działał tak samo lub lepiej niż BAP1. W pozostałych przypadkach przewagę wykazuje BAP1. Zauważa się, że procent przypadków, dla których pierwszy próg jest wartością optymalną maleje, przy danym n, ze wzrostem b /w skrajnym przypadku: $b=2^{31}-1$ i n=100, czas t=5,368 dla SIGMIN powstał wskutek 179 zmian progu/.

Przedstawione wyniki nie pozwalają wskazać algorytmu zdecydowanie lepszego, co poddaje w wyraźną wątpliwość konkluzję o supremacji algorytmu BASS przedstawioną w [2], gdzie punktem wyjścia była teza o dotychczasowej supremacji algorytmu BAP1.

Tabela 2

b	100			1000			n			n^2			$2^{31}-1$			
20	.000	.002	.003	.001	.002	.004	.001	.002	.003	.000	.002	.003	.001	.002	.003	B
	.001	.002	.006	.000	.004	.019	.001	.002	.003	.001	.003	.015	.000	.005	.024	S
40	.004	.008	.015	.004	.009	.014	.005	.007	.012	.004	.009	.015	.004	.009	.014	B
	.005	.009	.024	.006	.026	.142	.006	.007	.015	.006	.009	.174	.006	.043	.288	S
60	.007	.020	.042	.009	.021	.048	.007	.019	.046	.009	.021	.048	.009	.021	.050	B
	.014	.019	.034	.015	.036	.156	.013	.019	.034	.014	.061	.346	.013	.081	.498	S
80	.015	.035	.052	.017	.038	.065	.016	.037	.062	.017	.038	.065	.017	.038	.065	B
	.022	.030	.032	.026	.034	.127	.026	.031	.058	.027	.043	.336	.023	.056	.661	S
100	.024	.059	.101	.025	.042	.098	.024	.059	.101	.025	.072	.104	.025	.071	.104	B
	.042	.051	.092	.041	.072	.633	.040	.051	.092	.041	.306	3.374	.044	.453	5.368	S

Czasy są podawane w sekundach dla procesora centralnego. Przykładowo zapis: .034 oznacza części sekundy po przecinku.

Formułujemy tezę, iż jednostronny charakter używanych zadań oraz wiele nieuchwytnych lub trudno poddawalnych analizie czynników związanych z implementacjami orientowanymi na konkretny komputer, wykluczają możliwość stawiania kategorycznych ocen porównawczych dla algorytmów rozwiązujących /MZP/.

Stosowana obecnie metodologia porównań pozwala natomiast na stawianie stosunkowo wąskich ocen, każdorazowo związanych z wieloma omówieniami i uwarunkowaniami.

Zrealizowanie metodologii bardziej ~~uniwersalnej~~ napotyka na trudności teoretyczne i praktyczne. Jedno z możliwych wyjść, to próba ustalenia koszyka zadań testowych. Koszyk taki mógłby obejmować zadania losowe /przy założeniu różnych rozkładów prawdopodobieństw/, zadania o specjalnej strukturze uznane za trudne, oraz zadania powstałe z powielenia struktury zadań aplikacyjnych. Inne, bardziej teoretyczne ujęcie problemu mogłoby polegać na wykonywaniu różnorodnych testów z zastosowaniem aparatu statystycznego dla weryfikacji hipotez.

LITERATURA

[1] Borkowska H., Słomiński L.: Komputerowa realizacja maksiminowego zagadnienia przydziału. ZPM-79. Listopad, 1979, IBS, Warszawa.

[2] Carpaneto G., Toth: Algorithm for the Solution of the Bottleneck Assignment Problem. Computing v. 27, 1981 p. 179-187

[3] Derigs U.: On Three Basic Methods for Solving Bottleneck Transportation Problems. Naval Research Logistics Quart. v. 29 No. 3. 1982 p. 505-515

[4] Derigs U., Zimmermann U.: An Augmenting Path Method for Solving Linear Bottleneck Assignment Problems. Report 77-4 Mai 1977. Mathematisches Institut, Universität zu Köln /opublikowane także w Computing/.

[5] Ford L.R., Fulkerson D.R.: Flows in Networks. Princeton Univ. Press. Princeton N.J. 1962

[6] Finke G., Smith P.A.: Primal Equivalents to the Threshold Algorithm. Operations Research Verfahren v. 31, 1979 p. 185-198

[7] Garfinkel R.S.: An Improved Algorithm for the Bottleneck Assignment Problem. Operations Research v. 19, 1971 p. 1747-1751

[8] Gordon A.Ja.: Odin algorithm roszenija minimaksonoi zadaczi o naznaczenii. W: Issledowanija po diskretnoj optimizacii. Izd. Nauka, Moskwa 1976

[9] Gross O.: The Bottleneck Assignment Problem. Raport P-1630. RAND Corpor. 1959

[10] Pape U., Shön B.: Verwahren zur Lösung von Summen - and Engpass -Zuordnungsproblemen. Elektronische Datenverarbeitung v. 4, 1970 p. 149-163

[11] Słomiński L.: The Bottleneck Assignment Problem: An Efficient Algorithm. Paper presented at IX International Symposium on Mathematical Programming, Budapest, August 1976 /Arch.Autom. i Telemech. 1979 z.4/.

[12] Słomiński L.: An Efficient Approach to the Bottleneck Assignment Problem. Bulletin de l'Academie Polonaise des Sciences, Serie des sciences techniques. v. 25, 1977 no.4.

[3] Słomiński L.: Algorytmy rozwiązywania minimaksowego zagadnienia przydziału i ich komputerowe porównywanie. ZPM-7/83. Listopad 1983, IBS Warszawa.

[4] Zimmermann U.: Informacja prywatna, 1980

Recenzent: Prof.dr hab.inż.Antoni Niederliński

Wpłynęło do Redakcji do 30.03.1984r.

АЛГОРИТМЫ РЕШАЮЩИЕ МИНИМАКСНУЮ ЗАДАЧУ НАЗНАЧЕНИЯ И ИХ МАШИННОЕ СРАВНЕНИЕ

Р е з ю м е

В работе представлено схему , названную пороговой схемой , описывающую в сжатой и выгодной, с точки зрения сравнения, форме известные алгоритмы решения минимаксной задачи назначения. Даются примеры использования пороговой схемы. Анализируются результаты машинных сравнений алгоритмов и указываются некоторые недостатки и противоречия выводов , проводимых на основе этих результатов.

ALGORITHMS FOR SOLVING THE BOTTLENECK ASSIGNMENT PROBLEM AND THEIR COMPUTER BASED COMPARING

S u m m a r y

A treshold scheme for a unified and convenient from a comparative point of view description, of known algorithms for the bottleneck assignment problem, is introduced. Examples of application of the threshold scheme are described. Results of some computer comparisons of considered algorithms are discussed. Some inconsistences and false conclusions, usually derived from these results, are pointed out.