REKURENCYJNA METODA WYSZUKIWANIA WĄSKICH GARDEŁ W SYSTEMACH PRODUKCYJNYCH

Paweł Sewastianow, Waldemar Herka
sevast@icis.pcz.czest.pl
Instytut Informatyki Teoretycznej i Stosowanej
ul. Dąbrowskiego 73, 42-200 Częstochowa

Problem wąskiego gardła jest kluczowym zagadnieniem udoskonalania systemów produkcyjnych w tym: przy planowaniu produkcji, zwiększaniu wydajności systemu produkcyjnego oraz w ogóle przy optymalizacji działalności systemów technologicznych. W pracy opisano problem wąskich gardeł w systemach produkcyjnych w aspekcie jego formalizacji matematycznej jak również dokonano opisu merytorycznego. Wprowadzono nowe pojęcia służące określeniu wymogów i zależności technologicznych narzucanych na elementy systemu produkcyjnego reprezentowanego przez sieć. Analiza wymogów jest dokonywana na poziomie lokalnym (wyodrębniony ciąg technologiczny odpowiadający za jeden rodzaj produktu) jak i globalnym (cały system produkcyjny). Opracowano metodę lokalizacji wąskich gardeł w liniach produkcyjnych złożonych pod względem struktury zależności technologicznych (rozgałęzienia). Przedstawiono algorytmy zapisane w C++ ułatwiające implementację metody.

SŁOWA KLUCZOWE: wąskie gardło, systemy produkcyjne

1.WPROWADZENIE

Problem wąskiego gardła jest kluczowym zagadnieniem udoskonalania systemów produkcyjnych w tym: przy planowaniu produkcji (scheduling) [21], zwiększaniu wydajności systemu produkcyjnego oraz w ogóle przy optymalizacji działalności systemów technologicznych. Wszechobecność problemu wąskiego gardła spowodowała zainteresowanie nim specjalistów z różnych dziedzin, w tym i matematyków. Dzisiaj istnieje sporo różnych formalizowanych podejść do rozwiązania tego problemu.

W najbardziej ogólnej formie matematycznej problem wąskiego gardła może być sformułowany następująco: niech E= $\{e_1,e_2..e_m\}$ zbiór obrabiarek i F jest rodziną podzbiorów E. Wtedy dla dowolnego f $\in$F wydajność wąskiego gardła (minimalna wydajność e_i w f) może być przedstawiona jako cap(f,C) gdzie C wektor wydajności o wymiarze m. Wydajność rodziny F może być przedstawiona jako:

$$cap(F,C) = \max\{cap(f,C) \mid \forall f \in F\}$$

dla konkretnego wektora C zagadnienie optymalizacji systemu może być przedstawione w postaci

$$\max_{f \in F} \min_{e \in f} c(e) \qquad (1)$$

Zagadnienie (1) nazywa się dyskretnym problemem wąskiego gardła. Problem ten był intensywnie badany przez wielu autorów [2-8,10,11,13,14]. Klasyczna metoda - algorytm wartości granicznej, została pokazana w Edmonds i Fulkerson [8]. Od tego czasu, wiele wersji dla szczególnych przypadków, wariantów i uogólnień znalazło zastosowanie w optymalizacji kombinatorycznej. Typowymi przykładami są: zagadnienie przydziału wąskiego gardła [7,8,10,11],problem wąskiego gardła oparty na zadaniu drzewa częściowego [6,10], zagadnienie wąskiego gardła bazujące na matroidzie [2] i problem wąskiego gardła oparty o TSP [5], itd. W ostatnich latach optymalizacja wąskich gardeł została uogólniona

do leksykograficznego typu optymalizacji, zobacz Burkard i Rendl [4] i Sokkalingam i Aneja [14]. Rozszerzona forma problemu wąskiego gardła może być przedstawiona jako[20] :

$$\max \ cap(F,C) \tag{2}$$

$$\mathrm{W}^{\mathrm{T}}(C-\overline{C}) \le D, \tag{3}$$

$$C \ge \overline{C}, \tag{4}$$

gdzie $W=(w_1,w_2,..,w_m)^T$, w_i jest kosztem zwiększenia wydajności obrabiarki e_i na jednostkę, a $\overline{C}$ jest wektorem wydajności pierwotnej (przed optymalizacją).

Zagadnienie (2)-(4) wygląda atrakcyjnie, bo uwzględnia przy optymalizacji wydajności systemu koszty tej optymalizacji. Jednak w systemach rzeczywistych poważnym problemem jest lokalizacja wąskiego gardła w konkretnej części systemu. Rzecz w tym, że optymalizacja z zasady nie pozwala na usunięcie wąskich gardeł, bo wyeliminowanie jakiegoś wąskiego gardła w jednej części systemu powoduje, że wąskim gardłem staje się inna obrabiarka w innej części systemu. Dla technologów nie jest obojętne czy wąskie gardło znajduje się na początku czy na końcu łańcucha technologicznego. Przypuśćmy, że wąskim gardłem systemu jest pierwsza obrabiarka. Wtedy wszystkie następujące po niej powinny pracować z wydajnościami mniejszymi od możliwych, co powoduje marnowanie możliwości technologicznych całego systemu. Stąd wynika, że optymalizacja na podstawie (2)-(4) nie uwzględniająca nierównoważności znalezienia wąskiego gardła w różnych częściach systemu jest raczej zagadnieniem teoretycznym niż praktycznym. W praktyce zawsze istnieje problem znajdowania i lokalizacji wąskiego gardła z punktu widzenia istniejącego obrazu systemu idealnego odzwierciedlającego marzenie konstruktorów systemu produkcyjnego oraz technologów zajmujących się bezpośrednio jego eksploatacją. Z tego powodu w artykule przedstawiono pojęcie płynnego procesu produkcyjnego, który wygląda jako dostatecznie zrozumiały kompromis między wymogami technologicznymi i który można rozpatrywać w pewnym sensie jako model procesu „idealnego". Dalej problem poszukiwania wąskiego gardła rozwiązany jest w oparciu o porównywanie systemu rzeczywistego z systemem idealnym przechowującym struktury i wymogi technologiczne procesu rzeczywistego.

Reszta pracy realizowana jest w sposób następujący. W rozdziale 2 sformułowane jest pojęcie płynnie pracującego ciągu technologicznego, opisane są jego możliwe wady i zalety w aspekcie używania go do rozwiązywania problemów rzeczywistych, przedstawiona jest jego budowa i oparta na nim metoda lokalizacji wąskich gardeł w systemie. W rozdziale tym wprowadzono kilka nowych pojęć ułatwiających spójne sformułowanie opisywanej metody: **granula informacyjna**, **Wektor Lokalny**, **Waga Lokalna**, **Wektor Globalny**. Dodatkowo metodę opisano w sposób algorytmiczny umożliwiający niezbyt skomplikowaną implementację. W rozdziale 3 zawarte jest podsumowanie oraz perspektywy dalszych prac.

2. MODELOWANIE SYSTEMU IDEALNEGO

Jako idealny będziemy rozpatrywali płynnie pracujący ciąg technologiczny, który ma tylko jedno wąskie gardło w stanowisku końcowym. Innymi słowy

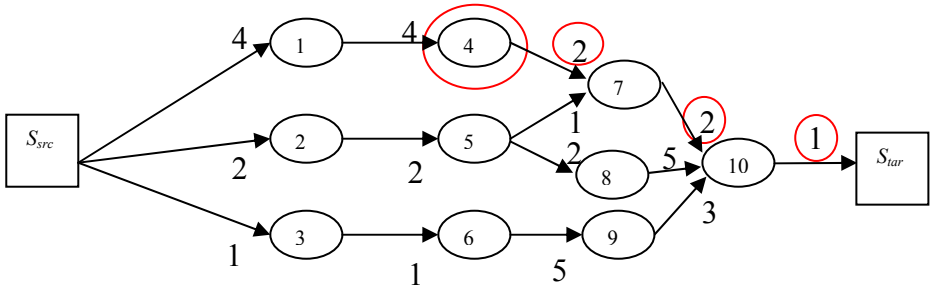
*Za **płynnie pracujący ciąg technologiczny** można uznać taki układ stanowisk, których wydajności pozwalają na sprostanie wymogom technologicznym, oraz w którym stanowisko końcowe pracuje ze swoją wydajnością paszportową – taki system pracuje niejako w takt stanowiska końcowego.*

Niezaprzeczalnie ciąg technologiczny działający w ten sposób posiada większą stabilność niż ciąg, w którym wąskie gardło znajduje się w innym miejscu. Główną przyczyną jest fakt, że jeśli stanowisko końcowe jest wąskim gardłem, to oznacza, że każde ze stanowisk, z którego składa się system posiada rezerwy mocy produkcyjnej, które gwarantują elastyczność powodującą zachowanie pewnego marginesu niepewności parametrów pracy stanowisk wprowadzonych np. przez czynnik ludzki. Dodatkowo stanowiska posiadające pewne nadwyżki mocy produkcyjnej nie są tak bardzo podatne na zużycie i awarie jak te pracujące „na najwyższych obrotach". Taki system jest także łatwiejszy w rozbudowie – modernizując stanowisko końcowe mamy gwarancję, że skutkiem będzie ogólna poprawa wydajności systemu. Osobną kwestią jest jak duża będzie to poprawa – decyduje o tym przede wszystkim wydajność stanowiska „najbliższego" wąskiemu gardłu oraz z jak dużą modernizacją mamy do czynienia.

2.1. Budowa płynnie pracującego ciągu technologicznego

Pierwszą czynnością jest zatem stworzenie modelu płynnie działającego systemu na bazie istniejących zależności technologicznych oraz wydajności stanowiska końcowego analizowanego podgrafu - na tym etapie nie jest konieczna znajomość wydajności paszportowych poszczególnych stanowisk. Istotą tego etapu jest znalezienie zależności determinującej liczbę półproduktów przypadających na jeden wyrób gotowy, wytwarzany przez stanowisko końcowe oraz odniesienie jej do wydajności stanowiska końcowego (według definicji ciągu pracującego w sposób płynny).

Opiszemy algorytm budowy płynnie pracującego ciągu technologicznego za pomocą prostego przykładu przedstawionego na rys.1, gdzie S_{src} S_{tar}.- magazyny źródłowy oraz końcowy, liczby około strzałek oznaczają **wymogi technologiczne**, obrazujące proporcje detali składających się na jednostkę detalu wyjściowego stanowiska. **Wymogi technologiczne** w formie tradycyjnej przedstawione są też przez macierz połączeń grafu (tablica 1).



Rys. 1. Podgraf pomocniczy do opisu budowy **Wektora Lokalnego**

Tablica 1. Macierz **C** połączeń do układu z rys.1.

	1	2	3	4	5	6	7	8	9	10	S_{src}	S_{tar}
1				4								
2					2							
						1						
4							2					
5							1	2				
6									5			
7										2		
8										5		
9										3		
10												1
S_{src}	4	2	1									
S_{tar}												

Krok 1: W celu zdefiniowania wymagań narzucanych na poszczególne stanowiska podgrafu w układzie działającym płynnie zdefiniujmy abstrakcyjną strukturę danych: **Wektor Lokalny**. Będzie on zawierał informacje pozwalające określić: stanowisko, którego dotyczy konkretny wpis, jego położenie w podgrafie oraz współczynnik określający wymagania narzucone przez stanowisko końcowe, które determinuje płynny przepływ w podgrafie. Wektor Lokalny będzie wektorem składającym się z wpisów nazywanych dalej **Wagą Lokalną.** Waga Lokalna przyporządkowana danemu stanowisku w podgrafie z kolei składa się z wpisów o pewnej strukturze, którą będziemy nazywać **granulą informacyjną.** Każda Waga Lokalna stanowiska posiada co najmniej jedną granulę informacyjną - w przypadku, gdy stanowisko ma tylko jednego następnika (odcinek liniowy – bez rozpływów). Liczba granuli informacyjnych zależy od liczby rozpływów na wyjściu stanowiska(na tym etapie brane pod uwagę są tylko rozpływy w ramach podgrafu). Poniżej, na przykładzie przedstawiona zostanie budowa granuli informacyjnej oraz zasada jej tworzenia.

Np. dla stanowiska siódmego granula informacyjna połączenia ze stanowiskiem dziesiątym będzie miała postać: {2,10,10} (odpowiednio dla połączenia stanowiska czwartego ze stanowiskiem siódmym : {4,7,10}). Sposób interpretacji wpisów na odpowiednich pozycjach granuli informacyjnej jest następujący:

- na pozycji pierwszej zapisuje się iloczyn współczynnika wymagań technologicznych rozpatrywanego stanowiska (zapisanego w macierzy połączeń) oraz współczynnika pobranego z pierwszej pozycji granuli informacyjnej dla stanowiska następnego – biorąc pod uwagę kierunek przepływu, a rozpatrywanego tuż przed bieżąco analizowanym stanowiskiem – jest to bardzo ważne szczególnie, gdy stanowisko posiada rozpływy tzn., że ma kilka następników – np. stanowisko piąte (wtedy kilka granuli informacyjnych przypada na jedną wagę lokalną danego stanowiska);
- pozycja druga zawiera indeks następnika związanego z daną granulą informacyjną;
- pozycja trzecia zawiera indeks stanowiska, które jest stanowiskiem końcowym w analizowanym podgrafie – informacja ta wydaje się nadmiarową na tym etapie, bo przecież w ramach podgrafu wszystkie granule informacyjne zawierają na trzeciej pozycji informację o tym samym stanowisku,

jednak jest bardzo przydatna kolejnym kroku algorytmu budowy płynnie działającego ciągu technologicznego, co zostanie pokazane poniżej.

Ponieważ procedura określania pozycji pierwszej granuli informacyjnej jest obliczana na zasadzie rekurencji, oczywistym jest określenie danej inicjującej ten proces. W procesach rekurencyjnych bardzo ważna jest kolejność rozpatrywania przypadków do obliczenia. Mając dany element i-ty potrafimy obliczyć i+1-szy. W omawianym algorytmie początek rozpatrywania znajduje się w stanowisku końcowym podgrafu, zatem należy z góry określić jego granulę informacyjną. Określmy wstępnie postać granuli informacyjnej identyfikująca połączenie stanowiska i ze stanowiskiem j:

$$\boldsymbol{GrInfo_{ij}[a_1,a_2,a_3]}, \tag{5}$$

gdzie: a_1 oznacza współczynnik wyliczany na podstawie informacji z poprzednio rozpatrywanych wag lokalnych, a_2 - indeks stanowiska połączonego ze stanowiskiem i, którego dotyczy granula poza nim samym (jest on określony poprzez j), a_3 indeks stanowiska końcowego w danym podgrafie.

W naszym konkretnym przykładzie na pierwszej pozycji Wektora Lokalnego (zawartość pierwszej Wagi Lokalnej) będzie granulą informacyjną zawierającą następujące dane: {1,10,10}. Charakterystycznym faktem jest, że Waga Lokalna stanowiska końcowego zawiera zawsze jedną granulę informacyjną, która nie jest przyporządkowana żadnemu konkretnemu połączeniu – wartością pierwszej pozycji tej granuli jest 1 (chodzi nam o ustalenie zależności odnoszącej się do wytworzenia jednego produktu przez stanowisko końcowe), a drugiej i trzeciej ta sama wartość - indeks stanowiska, któremu jest przyporządkowana ta Waga Lokalna, czyli stanowiska końcowego podgrafu odpowiadającego wypełnianemu Wektorowi Lokalnemu. Na podstawie tej informacji oraz wpisów w macierzy połączeń jesteśmy w stanie określić granulę informacyjną dla stanowisk:7, 8, 9, a z kolei na ich podstawie dla stanowisk:4, 5, 6 itd. aż do osiągnięcia stanowisk połączonych bezpośrednio z magazynem początkowym S_{src} (1,2,3). Na podstawie powyższego opisu słownego wprowadzamy formalny opis zawartości granuli informacyjnej, tzn. parametrów $a_{1,ij},a_{2,ij},a_{3,ij}$ dla połączenia stanowiska *i* ze stanowiskiem *j*:

$$a_{1,ij} = c_{ij} \cdot c_{jk} \cdot c_{kk1} \cdot,\ldots,\cdot c_{nn+1},\ a_{2,ij} = j,\ a_{3,ij} = S_{tar}, \tag{6}$$

gdzie: $c_{i,j}$ – współczynniki dla połączenia stanowisk z macierzy połączeń, wybrane w kolejności połączeń od stanowiska i do ostatniego n (w naszym przypadku, patrz rys.1, n=10). Oczywiście n+1 jest indeksem stanowiska końcowego podgrafu S_{tar}

Wektor Lokalny to rodzaj kontenera zawierającego granule informacyjne poszczególnych stanowisk podgrafu. Nie oznacza to jednak, że granule informacyjne są ulokowane w nim w sposób czysto przypadkowy: porządek rozmieszczenia ma bardzo duże znaczenie dla przebiegu obliczeń według algorytmu rekurencyjnego obliczania współczynnika na pierwszej pozycji odpowiedniej granuli informacyjnej. Z tego powodu wprowadzona została **Waga Lokalna**, która grupuje granule informacyjne w pakiet odnoszący się do danego stanowiska. Waga Lokalna obejmuje zatem informacje zawarte w granulach informacyjnych oraz kolejność umieszczenia jej w **Wektorze Lokalnym.** Informacja o kolejności umieszczania jest szczególnie ważna przy obliczaniu kolejnych granuli informacyjnych – bez takiego uporządkowania granuli informacyjnych mogłaby być trudniejsza budowa sensownego algorytmu obliczania kolejnych granuli

informacyjnych, potrzebnych do obliczenia płynnego przepływu. Cechą charakterystyczną jest to, że dzięki Wadze Lokalnej istnieje możliwość alokacji wielu egzemplarzy granuli informacyjnej na jednej pozycji **Wektora Lokalnego –** zależnie od ilości rozpływów danego stanowiska w ramach jednego podgrafu. Możliwość ta dodatkowo potwierdza konieczność wprowadzenia **Wagi Lokalnej**.

Opiszemy algorytm budowania Wektora Lokalnego podgrafu:

1) Alokujemy wektor na struktury: {indeks stanowiska, zbiór granuli informacyjnych} o rozmiarze równym liczbie stanowisk w podgrafie.

2) Wpisujemy granulę informacyjną do Wagi Lokalnej stanowiska końcowego (zawsze na pierwszej pozycji wektora).

3) Na podstawie Wagi Lokalnej stanowiska końcowego i wzoru na obliczenie granuli informacyjnej wypełniamy Wektor Lokalny .

Po wykonaniu kroków 1)-3) w rozpatrywanym przypadku otrzymamy **Wektor lokalny:**

numer stanowiska	10	7	4	1	5	2	8	9	6	3
Wagi_lokalne	{1,10,10}	{2,10,10}	{4,7,10}	{16,4,10}	{2,7,10}, {10,8,10}	{12,5,10}	{5,10,10}	{3,10,10}	{15,9,10}	{15,6,10}

Dodatkowych wyjaśnień wymaga sposób obliczenia Wagi lokalnej dla stanowisk: 5 i 2, ponieważ stanowisko 5 posiada rozpływy, a stanowisko 2 jest wyliczane na podstawie wagi lokalnej stanowiska 5. W tym przypadku, aby obliczyć Wagę Lokalną stanowiska 5 potrzebne są dwa kroki poprzedzające właściwe obliczenia: obliczenie Wagi Lokalnej stanowiska 7 oraz obliczenie Wagi Lokalnej stanowiska 8. Analogicznie do stanowiska 5, stanowisko 2 także potrzebuje dwóch kroków, ponieważ Waga Lokalna stanowiska 5, które poprzedza w obliczeniach stanowisko 2 zmienia się. Należy wziąć pod uwagę tego typu sytuacje i ilekroć zmienia się Waga Lokalna stanowiska od którego zależą obliczenia analizowanej Wagi Lokalnej, należy powtórnie przeprowadzać obliczenia. Dla Wagi Lokalnej stanowiska 2 nie dopisujemy kolejnej granuli informacyjnej, ponieważ posiada ono tylko jednego poprzednika (biorąc pod uwagę przepływ jest to następnik) .

Krok 2: Następnym etapem obliczania płynnego przepływu jest agregacja informacji uzyskanych w poprzednim etapie na poziomie poszczególnych podgrafów do struktury na poziomie całego systemu, którą nazwiemy **Wektorem Globalnym.** Jest to struktura o dostępie analogicznym do wektora (indeksowanym), a jej rozmiar jest równy liczbie stanowisk w układzie. Indeksy **Wektora Globalnego** odpowiadają indeksom stanowisk (inaczej niż w Wektorze Lokalnym). Poszczególne elementy Wektora Globalnego będziemy nazywać poprzez analogię do poprzedniego etapu **Wagą Globalną.** Waga Globalna podobnie jak Waga Lokalna grupuje granule informacyjne. Różnica polega na tym, że poszczególne Wagi Lokalne budowane są na poziomie podgrafu i z racji tego trzeci element każdej granuli informacyjnej wchodzącej w skład danej Wagi Lokalnej jest identyczny – równy indeksowi stanowiska końcowego podgrafu, którego dotyczy Wektor Lokalny. W Wektorze Globalnym w sytuacji, gdy stanowisko należy do kilku podgrafów, granule informacyjne Wagi Globalnej mogą się różnić pod kątem trzeciego elementu. Procedura wypełniania Wektora Globalnego nie jest skomplikowana - polega na przeglądaniu wszystkich Wektorów Lokalnych

i dopisywaniu do odpowiednich Wag Globalnych granuli informacyjnych zawartych w Wagach Lokalnych należących do stanowisk o takich samych indeksach (w numeracji układu, nie podgrafu). Algorytm w języku C++ przedstawiono w tablicy 2.

Tablica 2.Algorytm wyliczania wektora globalnego.

```
1 CalculateGlobalVect()
2 {
3 for( i=0 ; i < numberOfStages() ; i++ )
4       {
5       for( j= 0 ; j < numberOfSubGraphs() ; j++ )
6               {
7               GlobalVect[i]+=LocalVect[j]. find(i);
8               }
9       }
10 }
```

gdzie:

- **numberOfStages() –** podprocedura zwracająca liczbę stanowisk w układzie;
- **numberOfSubGraphs() -** podprocedura zwracająca liczbę podgrafów w układzie;
- **LocalVect[**j**]. find(**i**) –** metoda umożliwiająca znalezienie Wagi Lokalnej stanowiska i-tego w j-tym Wektorze Lokalnym (odpowiadającym j-temu podgrafowi) – jeśli Waga Lokalna nie zostanie znaleziona zwrócony zostanie stosowny kod uniemożliwiający błędne dopisanie(np.NULL);

Krok 3: Obliczenie wydajności dla układu o płynnym przepływie.

Na tym etapie dokonujemy wstępnej optymalizacji rozpływu na stanowiskach wspólnych dla kilku podgrafów oraz obliczamy wydajności stanowisk wymagane przez układ pracujący płynnie.

Stanowisko, od którego zależy więcej niż jeden podgraf posiada różne wymagania narzucane przez podgrafy, które je zawierają. Podgrafy traktowane autonomicznie nie przechowują w Wektorach Lokalnych, które je reprezentują informacji na temat wymagań innych podgrafów w stosunku do stanowisk wspólnych. Taką informację można zaobserwować w Wektorze Globalnym, który odwzorowuje cały układ. W celu optymalnego dobrania rozpływu należy określić porcję wymagań technologicznych stawianych przez dany podgraf wobec konkretnego stanowiska w stosunku do wymagań globalnych (na poziomie układu wymagania technologiczne stanowiska wspólnego są narzucane przez różne podgrafy). Można to zrealizować w sposób przedstawiony w tablicy 3.

Tablica 3.Obliczanie wydajności dla układu o płynnym przepływie

```
1 double  EvaluateChargeSharing( int i ,int subG)
2 {
3 double FullCharge = 0;
4 double PartialCharge =0;
5 double portion = 0;
6 for( int j = 0 ; j < numberOfGranule( GlobalVect[i] ); j++)
7         {
8         FullCharge+=GlobalVect[i].Granule[j][0] * avg( PassEff(GlobalVect[i].granule[j][2]) );
9
10        if( GlobalVect[i].granule[j][2] == finalStage(subG) )
11                PartialCharge += GlobalVect[i].Granule[j][0] ;
12        }
13  PartialCharge *= avg( PassEff( finalStage(subG) ) );
14
15  portion = PartialCharge/FullCharge;
16  return portion;
17 }
```

gdzie**:**

- parametry wejsciowe: **i** – indeks stanowiska; **subG** – indeks podgrafu, dla którego obliczamy rozpływ;

- GlobalVect[i].granule[j][2] oznacza pobranie trzeciego elementu j-tej granuli informacyjnej i-tej Wagi Globalnej(odpowiedzialnego za identyfikację podgrafu);

- numberOfGranule(GlobalVect[i]) funkcja zwracająca liczbę granuli informacyjnych w i-tej Wadze Globalnej;

- avg(PassEff(GlobalVect[i].granule[j][2])) oznacza uśrednienie wydajności paszportowej (jest to trafne w przypadku interwału zwykłego, jeśli chodzi o wydajności zapisane za pomocą interwałów rozmytych należy dokonać defuzyfikacji ; wydajności w postaci liczb rzeczywistych nie wymagają stosowania operacji uśredniania – jest ona neutralna w stosunku do tego typu danych) stanowiska określonego poprzez 2-gi element j-tej granuli informacyjnej i-tej Wagi Globalnej – co oznacza wydajność paszportową stanowiska końcowego podgrafu, z którego pochodzi dana granula informacyjna;

- finalStage(subG) oznacza funkcję zwracającą indeks stanowiska końcowego podgrafu subG.

Mając obliczoną porcję produkcji konkretnego stanowiska kierowaną do danego podgrafu jesteśmy w stanie obliczyć wydajność stanowiska wymaganą do tego, aby przepływ w układzie można było uważać za płynny. Będzie to wydajność sumaryczna, mówiąca o tym jaką wydajność w sumie musi mieć konkretne stanowisko, aby utrzymać płynny przepływ na skalę całego układu (lokalnie –biorąc pod uwagę przepływ w danym podgrafie wielkość ta może być identyczna – jeśli podgrafy są rozłączne lub mniejsza jeśli analizujemy stanowisko wspólne dla kilku podgrafów). Przy obliczeniach korzystamy z następującej zależności:

$$Wflex_i = \sum_{j=0}^{n} \lambda_{ij} \bullet Wk_j \qquad (3)$$

gdzie: $Wflex_i$ – wydajność stanowiska i w płynnym przepływie; λ_{ij} – porcja produkcji stanowiska i-tego kierowana do podgrafu j (obliczona na podstawie procedury **EvaluateChargeSharing**); n – liczba podgrafów, które zależą od stanowiska i; Wk_j – wydajność stanowiska końcowego podgrafu j.

2.2. Szukanie wąskich gardeł systemu

Po zbudowaniu systemu płynnie działającego, jego parametry (wydajności) na poszczególnych stanowiskach porównujemy z ich danymi paszportowymi. Operację porównania można przeprowadzać na poziomie globalnym lub na poziomie lokalnym.

Na poziomie globalnym porównujemy sumaryczne wymagania obliczane według sposobu przedstawionego powyżej (Wflex) z wydajnością paszportową stanowiska. W ten sposób uzyskujemy informację jakościową o stosunku wymagań układu o płynnym przepływie do wydajności paszportowych (Wpass), tzn. wynik porównania typu boolowskiego odczytujemy jako „niedobór" (Wflex>Wpass), „nadwyżka" (Wflex<Wpass) lub „idealne dopasowanie" (Wflex=Wpass) wydajności paszportowej i wymaganej przez płynny przepływ. Oznacza to np. że stanowiska przy, których powyższe porównanie dało w rezultacie niedobory to potencjalne wąskie gardła. Na tym się jednak kończy, bowiem nie można określić tego dokładnie, które ze stanowisk jest faktycznym wąskim gardłem jeśli wystąpi kilka z „niedoborem" – wszystkie są jednakowo traktowane(operatory boolowskie zwracają tylko dwie wartości: 0 i 1). Poza tym jeśli stanowisko z „niedoborem" jest wspólne nie można nawet określić podgrafu, w którym mogło by być wąskim gardłem – ze względu na analizę na podstawie wielkości sumarycznych wszystkie podgrafy, w których skład wchodzi dane stanowisko są jednakowo podejrzewane o wystąpienie wąskiego gardła. Sytuacja jest nieco prostsza choć niewystarczająco jasna dla układów rozłącznych – ze względu na brak współdzielenia stanowisk nie ma dylematu: na który podgraf „niedobór" będzie miał większy wpływ. Można zatem stwierdzić, że podejście globalne jest przydatne jedynie przy czysto pobieżnej ocenie układu jako całości – wyznaczenia jego słabszych punktów – bez szczegółów ilościowych.

Porównywanie na poziomie lokalnym – na poziomie podgrafów jest nieco bardziej złożone – co za tym idzie bardziej pracochłonne. Rezultaty otrzymane w wyniku takiego postępowania dają jednak bardziej kompletne w sensie ilościowym rezultaty. Przeprowadzając takie porównanie należy skorzystać z porcji produkcji kierowanej do danego podgrafu - nie bierzemy pod uwagę wydajności sumarycznej, a jedynie wydajność wymaganą przez płynny przepływ w danym podgrafie – analogicznie jest z wydajnością paszportową stanowiska. Poniżej przedstawiony jest algorytm (Tablica 4.) wyszukiwania wąskiego gardła w podgrafie.

Tablica 4. Algorytm wyszukiwania wąskiego gardła w podgrafie.

```
1 int  FindBottleneck( i )
2 {
3 vector<EffType> diff(sizeOf(i));
4 for( int j = 0 ; j < sizeOf(i); j++)
5         {
6         EffType SumEffFlex = 0; EffType  LocalDiff = 0;
7         for( int k = 0 ;  k < numberOfConsumers(j) ; k++)
8                 {
9                 SumEffFlex += EvaluateChargeSharing(j,k) *EFlex(j)* PassEff( finalStage( k ) ) ;
10                }
11        EffType EffPass = EvaluateChargeSharing(j,k) * PassEff(j) ;
12        LocalDiff = EffPass – SumEffFlex;
13        diff[j] = LocalDiff / ( EvaluateChargeSharing(j,k) * PassEff( finalStage( k ) ) ;
14        }
15  EffType minval = diff[0];
16  int stage_index = 0;
17 for( int l = 0 ; l < sizeOf(i); l++)
18        {
19        if(diff[l]<EffType_cast(0))
20                if(diff[l]<minval)
21                        {
22                        minval = diff[l];
23                        stage_index = l;
24                        }
25        }
26 return globalIndex(stage_index,i);
27 }
```

gdzie**:**

- parametr wejściowy **i** – indeks podgrafu, w którym szukane jest wąskie gardło;
- zwracany rezultat: indeks stanowiska (w numeracji globalnej) stanowiącego wąskie gardło podgrafu;
- **vector<EffType> diff(sizeOf(i))** – zapis oznaczający alokację wektora elementów typu EffType (wydajności mogą być różnego typu – jest to zapis abstrakcyjny) o razmiarze wyliczonym z wielkości podgrafu - **sizeOf(i);**
- **numberOfConsumers(j)** – funkcja zwracająca liczbę konsumentów – podgrafów zależnych od stanowiska j;
- **PassEff(finalStage(k))** – pobranie wydajności paszportowej stanowiska końcowego podgrafu k ;
- **EFlex(j)** – funkcja obliczająca sumę wymagań technologicznych (na jednostkę produktu gotowego) na podstawie danych zawartych w granulach informacyjnych wagi globalnej stanowiska j ;
- **globalIndex(stage_index,i)** – wyliczenie indeksu globalnego na podstawie indeksu lokalnego oraz indeksu podgrafu;

Algorytm można podzielić na dwie zasadnicze części:

1) wypełnianie wektora różnic (linie: 4 - 14);

2) obliczanie maksymalnej różnicy (linie: 17 –25)

Pierwsza część składa się z następujących operacji (przeprowadzanych w pętli dla wszystkich elementów podgrafu):

- obliczenie wydajności wymaganej przez układ o płynnym przepływie (poprzez użycie funkcji **EvaluateChargeSharing** ograniczamy się do wymagań stawianych przez dany podgraf) (linie: 7 – 10);
- obliczenie części wydajności paszportowej stanowiska trafiającej do danego podgrafu (linia : 11);

- obliczenie różnicy wydajności paszportowej i układu w płynnym przepływie – określa niedobór mocy produkcyjnej (< 0) lub jej zapas (> 0) w jednostkach elementów wytwarzanych przez stanowisko (tzw. różnica lokalna) (linia : 12) ;
- obliczenie tzw. różnicy globalnej (niedobór/zapas mocy produkcyjnej odniesiony do całości produkcji stanowiska) i przypisanie jej do wektora na pozycję odpowiadającą pozycji stanowiska w wektorze lokalnym (podgrafie);

Druga część przebiega według standardowego algorytmu wyszukiwania minimum. Porównywanie nie obejmuje jednak wszystkich elementów wektora różnic, a jedynie te które wykazują niedobór mocy produkcyjnej (<0). Powstaje jednak pytanie : Jak traktować zapis „ < 0 ” w warunkach niepewności, kiedy mamy do czynienia z danymi w postaci interwałów lub interwałów rozmytych ? Odpowiedź jest dosyć intuicyjna – bierzemy pod uwagę każdą sytuację, w której niedobór może wystąpić, a więc wszystkie różnice których co najmniej część jest „<0”. Oznacza to w przypadku interwału, że co najmniej jego początek leży po lewej stronie zera. Znalezione minimum różnic jest maksymalnym niedoborem mocy produkcyjnej, a stanowisko mu odpowiadające wąskim gardłem systemu. Może zdarzyć się sytuacja, w której wszystkie różnice będą dodatnie – jest to sytuacja wysoce komfortowa – mamy wtedy do czynienia z podgrafem pracującym w warunkach płynnego przepływu.

Należy dodać, że wąskich gardeł w całym układzie może być w skrajnym wypadku tyle, na ile podgrafów jest on podzielony. Istnieją jednak sytuacje, w których wąskie gardło może być wspólne – wtedy ich liczba może być mniejsza (nie dotyczy to układów rozłącznych).

3. PODSUMOWANIE

Analiza problemu wąskich gardeł nie może być bagatelizowana. Przypuśćmy, że jeśli nawet prawie cały system składa się z elementów spełniających najwyższe normy wydajności oraz stabilności, wystarczy jeden kluczowy element o wydajności niższej niż pozwalają na to inne składniki, a globalna wydajność sytemu jest zdeterminowana przez wydajność tego najmniej efektywnego. Rozbudowa systemu bez kontroli jego wąskich gardeł może nie przynieść żadnych wymiernych rezultatów poza samymi kosztami rozbudowy. Każda optymalizacja systemu produkcyjnego powinna zakładać istnienie wąskich gardeł i uwzględnienie ich wpływu na system. Opisana w niniejszej pracy metoda pozwala kontrolować system pod względem lokalizacji w nim miejsc stanowiących jego wąskie gardła. Każda optymalizacja systemu powinna zaczynać się od tych miejsc. Mamy wtedy gwarancję, że będzie na pewno skuteczna.

Dalsze prace będą ukierunkowane na wykorzystanie opisanej metody do modelowania systemu oraz podczas procesu planowania produkcji (scheduling) w warunkach niepewności.

LITERATURA

[1] Ahuja RK, Magnanti TL, Orlin JB. Network flows. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[2] Averbakh I, Berman O, Punnen AP. Constrained matroidal bottleneck problems. Discrete Applied Mathematics 1995;63:201-14.
[3] Berman O, Einav D, Handler G. The constrained bottleneck problem in networks. Operations Research 1990;38:178-81.
[4] Burkard RE, Rendl F. Lexicographic bottleneck problems. Operations Research Letters 1991;10:303-8.
[5] Burkard RE, Sandholzer W. Efficiently solvable special cases of bottleneck traveling salesman problem. Discrete

Applied Mathematics 1991;32:61-76.
[6] Camereni PM. The minimax spanning tree problem and some extensions. Information Processing Letters 1978;7:10-4.
[7] Derigs U, Zimmermann U. An augmenting path method for solving linear bottleneck assignment problems. Computing 1978;19:285-95.
[8] Edmonds J, Fulkerson DR. Bottleneck extrema. Journal of Combinatorial Theory 1970;8:299-306.
[9] Fulkerson DR, Harding GC. Maximizing the minimum source-sink path subject to a budget constraint. Mathematical Programming 1977;13:116-8.
[10] Gabow HN, Tarjan RE. Algorithms for two bottleneck optimization problems. Journal of Algorithms 1988;9:411-7.
[11] Garfinkel R. An improved algorithm for the bottleneck assignment problem. Operations Research 1971;19:1747-51.
[12] Papadimitriou CH, Steiglitz K. Combinatorial optimization: algorithms and complexity. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[13] Punnen AP, Nair KPK, Aneja YP. Generalized bottleneck problems. Optimization 1995;35:159-69.
[14] Sokkalingam PT, Aneja YP. Lexicographic bottleneck combinatorial problems. Operations Research Letters 1998;23:27-33.
[15] Yang C, Zhang J. Inverse maximum capacity problems. OR Spektrum 1998;20:97-100.
[16] Yang C, Zhang J. Two general methods for inverse optimization problems. Applied Mathematics Letters 1999;12:69-72.
[17] Zhang J, Ma Z. A network flow method for solving some inverse combinatorial optimization problems. Optimization 1996;37:59-72.
[18] Zhang J, Liu Z. A further study on inverse linear programming problems. Journal of Computational and Applied Mathematics 1999;106:345-59.
[19] Zhang J, Liu Z, Ma Z. Some reverse location problems, European Journal of Operations Research, to appear.
[20] Zhang J, Yang C, Lin Y. A class of bottleneck expansion problems, Computers & Operations Research 2001;505:519-28
[21] Cristofari M, Caron F, Tronci M. Dynamic scheduling approach in case of machine breakdown and preventive maintenance - 2000 Summer Computer Simulation Conf.