na prawach rękopisu

INSTYTUT CYBERNETYKI TECHNICZNEJ
POLITECHNIKI WROCŁAWSKIEJ

RAPORT SERII PREPRINTY NR 34/80

OPTYMALIZACJA ROZDZIAŁU ZASOBOW W KOMPUTEROWYM SYSTEMIE STERUJĄCYM Z UWZGLĘDNIENIEM JAKOSCI STEROWANIA

/rozprawa doktorska/

Wojciech Myszka

Promotor

Prof. ZDZISŁAW BUBNICKI

Słowa kluczowe: system operacyjny, rozdział zasobów, proces technologiczny, sterowanie.

WROCŁAW 1980

Wydawnictwo Politechniki Wrocławskiej

SPIS TRESCI

1. Problemy rozdziału zasobów w Komputerowych Systemach Sterujących. 1
1.1 Wstęp. 1
1.2 Podstawowe pojęcia i określenia używane w pracy. 7
1.3 Formalne postawienie problemu pracy. 16
2. Rozdział zasobów w Komputerowym Systemie Sterującym obiektami statycznymi. 20
2.1 Rozdział zasobu limitującego wykorzystanie Systemu Cyfrowego. 20
2.1.1 Własności optymalnego rozdziału zasobów. 20
2.1.2 Opis algorytmu rozdziału zasobów. 27
2.2 Rozdział wielu zasobów. 32
2.2.1 Własności zadania rozdziału wielu zasobów 32
2.2.2 Algorytm optymalnego rozdziału wielu zasobów. 33
3. Rozdział zasobów w Komputerowym Systemie Sterującym obiektami dynamicznymi. 48
3.1 Zadanie rozdziału jednego zasobu. 48
3.1.1 Podstawowe własności zadania. 48
3.1.2 Algorytm rozdziału zasobów oparty na metodzie podziału i ograniczeń. 50
3.2 Zadanie rozdziału wielu zasobów. 60
3.2.1 Własności rozwiązań optymalnych. 60
3.2.2 Metoda konstruowania algorytmu. 61
4. Uogólnienia. 63
4.1 Różne okresy aktywizacji programów. 63
4.2 Różne linie krytyczne. 69
4.2.1 Rozdział jednego zasobu. 69
4.2.2 Rozdział wielu zasobów. 74
5. Sposób realizacji algorytmów w Komputerowych Systemach Sterujących. 78
5.1 Rozwiązywanie zadania optymalizacji na bieżąco. 78
5.2 Realizacja algorytmów rozdziału zasobów w układzie zamkniętym. 79
5.3 Sterowanie rozdziałem zasobów w układzie zamkniętym. 81
6. Ilustracja zadania rozdziału zasobów na przykładzie KSS konwersją SO_2. 83
7. Zakończenie. 88

W pracy postawiono i rozwiązano problem rozdziału zasobów, w pewnej klasie Komputerowych Systemów Sterujących procesami technologicznymi, optymalnego ze względu na jakość sterowania. Zakłada się, że każdy z programów wydających sterowania określony ma termin, przed którym zakończone być muszą obliczenia. W przypadku gdy program nie może zakończyć obliczeń w terminie zostaje zaniechany, a na obiekt wydawane jest sterowanie zastępcze. Omówiono algorytmy rozdziału zasobów w takim Systemie Cyfrowym w różnych przypadkach.

Opracowano oryginalny algorytm szeregowania programów przed liniami krytycznymi uwzględniający dodatkowe zasoby.

1. PROBLEMY ROZDZIAŁU ZASOBÓW W KOMPUTEROWYCH SYSTEMACH STERUJĄCYCH.

1.1. Wstęp.

Praca dotyczy problematyki rozdziału zasobów w Komputerowych Systemach Sterujących. Rozdział zasobów jest podstawową funkcją każdego Systemu Operacyjnego. Ma on podstawowe znaczenie dla efektywności wykorzystania Systemu Cyfrowego. Stąd też ogromna rola zadań projektowania i budowy Systemów Operacyjnych oraz duża liczba opracowań zarówno teoretycznych jak i praktycznych.

Systemy Cyfrowe pracujące w czasie rzeczywistym stanowią obiekt szczególnego zainteresowania. Odnosi się ono zwłaszcza do Komputerowych Systemów Sterujących.

W większości dotychczasowych opracowań otoczenie Systemu Cyfrowego /SC/ - sterowany proces technologiczny traktowany był w sposób drugorzędny. Stawiane zadania rozdziału zasobów nie uwzględniały własności obiektu sterowania, a jedynie jego pewne czasowe charakterystyki: parametry strumieni zgłoszeń, ich liczbę, czas obliczeń, wartości linii krytycznych, itp.

Zadania rozdziału zasobów w KSS powinny być traktowane jako zadania szeregowania zbioru programów z uwzględnieniem kosztów za niewykonanie ich w terminie. Ten problem, podobnie jak większość zagadnień szeregowania z uwzględnieniem kosztów ma niezbyt bogatą literaturę [35] .

Rozpatrywane najczęściej zadania szeregowania, np.: [6,7,9,17,20,23,31,33,35,71] jako kryterium szeregowania wykorzystują: czas trwania obliczeń zbioru programów, średni czas przepływu, maksymalne lub średnie opóźnienie programów, terminowość zakończenia obliczeń.

Algorytmy rozwiązywania opisanych wyżej zadań są wykorzystywane do projektowania SO czasu rzeczywistego, pomimo, że tak formułowane kryteria są typowe dla projektowania SO do przetwarzania typu wsadowego.

Szeroki przegląd zagadnień szeregowania programów na wielu procesorach zawarty jest w pracach [6,8,9,20] . Stosunkowo niewiele prac poświęconych jest zagadnieniom szeregowania programów z uwzględnieniem dodatkowych zasobów. Najczęściej rozpatrywane są szczególne przypadki rozdziału jednego dodatkowego zasobu, bądź jednakowych żądań zasobów. Najpełniejsze ujęcie problemów szeregowania z uwzględnieniem wielu zasobów zawiera praca Goetza [30] .

Problemem typowym dla Komputerowych Systemów Sterowania jest zadanie szeregowania programów przed liniami krytycznymi. Szereg ciekawych rezultatów dotyczących tego zagadnienia znaleźć można w [5,6,7,9,20,49] . Podczas rozwiązywania tego typu zadań należy określić czy istnieje uszeregowanie programów przed liniami krytycznymi /tzw. uszeregowanie dopuszczalne/, a następnie znaleźć je.

W przypadku, gdy nie istnieje uszeregowanie dopuszczalne, problem pozostaje otwarty. Projektant SO rozważyć może kilka wariantów postępowania:

- zwiększyć liczbę tych zasobów SC, które ograniczają jego przepustowość,
- określić jakie straty powstaną w przypadku nieterminowego kończenia programów i wybrać /zaprojektować/ taką procedurę szeregowania, która będzie je minimalizować,
- w przypadku, gdy programy nie mogą być kończone po upływie terminu /systemy typu hard - real - time/, określić należy sposób wydawania decyzji zastępczych oraz straty jakie powstaną z tego tytułu i wybrać /zaprojektować/ taką procedurę, która je minimalizuje.

Prace Gościńskiego [32] i Gościńskiego i Zielińskiego [33] rozpatrywały problemy, które można zaklasyfikować do drugiej z wymienionych metod postępowania. W [32] podano, między innymi, ogólne uwagi na temat doboru wskaźników efektywności działania SC uwzględniające efektywność wykorzystania zasobów, koszt realizacji algorytmów SO oraz ich wpływ na jakość sterowania. W [33] rozwiązano tak postawiony problem w pewnym, szczególnym przypadku, badając wpływ częstości aktywizacji programów

na jakość sterowania i koszty jego realizacji.

Jak można wywnioskować z przeglądu literatury, problematyka szeregowania zadań i rozdziału zasobów uznawana jest powszechnie za najważniejszą dla rozwoju praktyki i teorii projektowania Systemów Operacyjnych. Doceniana jest również waga tego zagadnienia w odniesieniu do SO KSS. Zbyt mało poświęca się miejsca specyfice takich zastosowań elektronicznych maszyn cyfrowych. W szczególności zadania projektowania algorytmów sterowania procesem technologicznym oraz algorytmów rozdziału zasobów SC traktowane są osobno. Powstaje zatem potrzeba wspólnego rozpatrywania obydwu problemów, tym bardziej, że w cyfrowych systemach sterujących powiązania pomiędzy SO a programami użytkowymi są bardzo silne.

Przedstawiana praca stanowi pewną próbę wspólnego traktowania obu zadań. Poświęcona jest ona zadaniom szeregowania programów i rozdziału zasobów w przypadku, gdy nie wszystkie programy mogą być wykonane przed upływem linii krytycznych. Wybrano ostatni z opisanych wcześniej sposobów postępowania. Rozpatrzono typowe zadania sterowania procesami technologicznymi złożonymi z obiektów statycznych albo dynamicznych. Sterowania wydawane są na obiekty procesu technologicznego cyklicznie, ze stałym okresem. W trakcie obliczeń, każdy z programów wydających sterowania, korzysta z zasobów SC. Zapotrzebowanie na zasoby może zmieniać się w trakcie obliczeń. Założono, że zarówno czas obliczeń, jak i zapotrzebowanie na zasoby są znane. SO podejmuje w każdym takcie swojej pracy decyzje dotyczące rozdziału zasobów i uruchamia programy, których zapotrzebowanie na zasoby może być spełnione. Pozostałe programy oczekują na przydział zasobów.

Po przerwaniu każdy z programów jest kontynuowany od miejsca zawieszenia. W chwili gdy ma być wydane sterowanie na obiekt, SO sprawdza czy program wyliczający je zakończył obliczenia; jeżeli tak, na obiekt wydawana jest wyliczona wartość sterowania; w przeciwnym przypadku podawana jest, przygotowana wcześniej wartość zastępcza. W ten sposób, ponieważ sterowania zastępcze różnią się od wartości optymalnych, następuje pogorszenie jakości sterowania procesem. Decyzje SO powinny

być podejmowane tak, aby pogorszenie jakości sterowania było minimalne. Powyższy problem w sposób formalny postawiony zostanie w rozdziale 1.3.

Przyjęcie deterministycznego opisu SC i programów jest pewnym uproszczeniem rzeczywistości jednak jest możliwe do przyjęcia w przypadku KSS, gdzie sprzęt jest niezawodny, zestaw programów ustalony, a ich charakterystyki dobrze znane.

Wydawać się może, że problemy wyboru decyzji SO i sterowań oddziaływujących na obiekt dynamiczny powinny być rozwiązywane wspólnie, przy założonej strategi sterowań zastępczych.
Takie postępowanie nie jest możliwe z następującego powodu: we wszystkich rozważaniach potrzebna jest znajomość czasu obliczeń programów. Może być on określony dopiero po uzyskaniu algorytmu sterowania. W przypadku wspólnego rozpatrywania obu problemów algorytm wyznaczania sterowań nie jest znany - nie można podać czasu jego realizacji.

Celem pracy jest opracowanie metodyki projektowania i algorytmów rozdziału zasobów w Systemie Operacyjnym KSS, Optymalnych ze względu na jakość sterowania, analiza możliwości praktycznej realizacji otrzymanych algorytmów w KSS i porównanie ich z priorytetowymi metodami szeregowania.

Wyniki uzyskane w pracy można podsumować w następujący sposób:

- sformalizowano opis programu i Systemu Cyfrowego oraz opracowano i postawiono zadania optymalizacji zasobów w Komputerowych Systemach Sterujących,
- opracowano łatwy w realizacji algorytm rozdziału jednego zasobu w KSS obiektami statycznymi oraz oryginalny algorytm rozdziału wielu zasobów oparty na zadaniu poszukiwania przepływu o minimalnym koszcie w specjalnie skonstruowanej sieci,
- zaproponowano algorytmy oparte na metodzie podziału i ograniczeń rozdziału jednego i wielu zasobów w KSS obiektami dynamicznymi,

- podano algorytm szeregowania programów oparty o zadanie programowania liniowego w przypadku gdy każdy z programów ma inną linię krytyczną,
- zaproponowano metodykę konstruowania algorytmów rozdziału zasobów w przypadku, gdy częstość aktywizacji każdego z programów jest inna,
- przeanalizowano sposób i możliwości realizacji algorytmów optymalnego rozdziału zasobów w KSS,
- zbadano symulacyjnie niektóre z otrzymanych algorytmów,
- zilustrowano proponowane podejście na przykładzie pewnego zadania sterowania procesem konwersjii SO_2.

Zawartość pracy jest następująca: w dalszej części rozdziału pierwszego zdefiniowane zostały podstawowe pojęcia używane w całej pracy, opisano zadania sterowania optymalnego procesem technologicznym oraz algorytmy ich realizacji. W końcowej części rozdziału postawiono w sposób formalny zadania optymalnego rozdziału zasobów rozwiązywane w pracy.

Rozdziały drugi i trzeci poświęcone są rozwiązaniu postawionych zadań w przypadku KSS obiektami statycznymi i dynamicznymi. Układ obu rozdziałów jest podobny: najpierw rozpatrywany jest przypadek rozdziału jednego, a następnie wielu zasobów. Każde z rozwiązywanych zadań opisane jest w trzech etapach: postawienie zadania i własności rozwiązań optymalnych, algorytm jego rozwiązywania oraz algorytm rozdziału zasobów SO KSS.

W rozdziale czwartym pokazano sposób uogólnienia rozwiązywanych problemów. Rozpatrzono dwa przypadki: gdy okresy aktywizacji programów są różne oraz gdy okresy aktywizacji są jednakowe, ale różne są terminy przed którymi programy muszą zakończyć obliczenia. Sposób prezentacji obu zadań jest jednakowy jak w rozdziałach 2 i 3.

W rozdziale piątym omówiono trzy zasadnicze sposoby realizacji algorytmów rozdziału zasobów:

- rozwiązanie zadania optymalizacji na bieżąco /tzn. decyzje podejmowane są na podstawie rozwiązania odpowiedniego zadania optymalizacji/,

- w układzie zamkniętym /tzn. decyzje podejmowane są na podstawie pomiarów parametrów obiektu/,
- w układzie otwartym /tzn. wcześniej wyliczone decyzje są odtwarzane/,

oraz możliwości stosowania każdego z powyższych sposobów realizacji.

W rozdziale szóstym metodykę podejścia zaproponowanego w pracy zilustrowano na przykładzie zadania sterowania procesem produkcji kwasu siarkowego metodą kontaktową.

Rozpatrywany przykład, ze względu na szereg uproszczeń ma znaczenie ilustratywne i służy jedynie do zaprezentowania pewnych idei.

Na zakończenie, w rozdziale siódmym podsumowano wyniki osiągnięte w pracy oraz wskazano na możliwości i kierunki dalszych badań.

Uzupełnieniem pracy jest dodatek [56] zawierający opracowane programy w języku FORTRAN maszyn serii ODRA 1300. W pracach [10,11,15,36] przedstawione są praktyczne realizacje Systemów Operacyjnych dla potrzeb sterowania procesami chemicznymi i eksperymentem.

Rozprawa powstała jako fragment badań dotyczących Systemów Operacyjnych Czasu Rzeczywistego, prowadzonych w Zespole Systemów Sterowania, pod kierunkiem Z.Bubnickiego. Innymi pracami o zbliżonej tematyce są [11,12,25,38,43,44,53,55] , dotyczą one szeroko rozumianego problemu projektowania Systemów Operacyjnych.

Autor pragnie wyrazić serdeczne podziękowanie Promotorowi pracy, prof. dr hab.inż. Zdzisławowi Bubnickiemu za opiekę naukową oraz szereg cennych wskazówek; kolegom z Zespołu Systemów Sterowania Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, a szczególnie dr inż.Ewarystowi Rafajłowiczowi za dyskusje i cenne uwagi pod adresem pracy oraz kol.kol. mgr inż.Leszkowi Borzemskiemu, mgr inż.Henrykowi Kordeckiemu i mgr inż.Andrzejowi Staniszowi, z którymi autor spędził wiele czasu na ciekawych i bardzo pożytecznych dyskusjach.

1.2. Podstawowe pojęcia i określenia używane w pracy

Poniżej podane zostaną określenia podstawowych pojęć /takich jak System Cyfrowy, Komputerowy System Sterowania, System Operacyjny, Zasób, itp/ używanych w tej pracy, w zakresie niezbędnym do prowadzenia rozważań. Nie będą to definicje, a jedynie określenia mające na celu sprecyzowanie zakresu pojęciowego używanych terminów.

Określenie 1 System Cyfrowy /SC/ rozumiany będzie jako zbiór zasobów z Systemem Operacyjnym oraz programami użytkowymi.

Według powyższego określenia na SC składają się procesory centralne, pamięć operacyjna, urządzenia wejścia/wyjścia, pamięci zewnętrzne, układy sprzeżenia z obiektem, zasilacze, itp. czyli tzw. sprzęt, oraz oprogramowanie: System Operacyjny, programy organizacyjne oraz programy użytkowe.

Dalej zajmiemy się dokładniej zasobami SC. Najważniejszą rolę wśród zasobów odgrywa procesor centralny - wszystkie programy użytkowe oraz System Operacyjny do swej pracy potrzebują czasu jednostki centralnej. Jest to też jeden z najbardziej kosztownych zasobów. Innymi zasobami są np.: urządzenia WE/WY oraz zbiory danych w pamięciach zewnętrznych, a także kody procedur bibliotecznych pamiętanych w P.O. Wszystkie wymienione zasoby mają następującą wspólną cechę - nie zużywają się, a po wykorzystaniu są zwracane do puli zasobów i mogą być wykorzystane przez inne programy. Tej grupie zasobów nadamy nazwę zasobów niezużywalnych /wielokrotnego użytku, ang.: serially reusable resources/.

W pracy nie będą rozpatrywane tzw. zasoby zużywalne /ang.: consumable resources/. Tego typu zasobami są np. wiadomości /ang. message/; są one tworzone przez programy - nadanie wiadomości, a następnie zużywane przez inny program - odebranie wiadomości. Powyższy podział zaczerpnięto z pracy Show'a [65].

Zbiór wszystkich zasobów podzielimy na klasy równoważności. Do jednej klasy zaliczymy wszystkie te zasoby, które mogą się wzajemnie zastępować.

Każdą klasę nazywać będziemy typem lub rodzajem zasobu. Załóżmy, że w rozpatrywanym SC jest r rodzajów zasobów. Niech $c^{(j)}$; j=1,2,...,r, oznacza liczność j-tej klasy /rodzaju/ zasobów.
Wektor $c = [c^{(1)}, c^{(2)}, \ldots, c^{(r)}]^T$ określa liczbę zasobów w SC.

Wszystkie rozważania w pracy prowadzone są przy założeniu niezmienności liczby zasobów w czasie. Wiele rozważań można bezpośrednio rozszerzyć na przypadek zmiennej liczby zasobów /np. w rozdz. 2.1./. W rozdziale 5 podano w jaki sposób rozwiązać postawiane zadania przy zmienności zasobów w czasie. W pracy rozpatrywana jest szczególna klasa SC tzn. Systemy Cyfrowe Czasu Rzeczywistego /lub inaczej Systemy pracujące w Czasie Rzeczywistym/. Obejmują one szeroką klasę systemów o działaniu bezpośrednim pracujących w środowisku o określonych wymaganiach na czas reakcji [70] . Do grupy tej zaliczyć można systemy rezerwacji biletów lotniczych, systemy sterowania procesami technologicznymi a także specjalizowane Systemy Cyfrowe pracujące w komputerowych sieciach obrony przeciwlotniczej [51,52] .

Poniżej podane określenie charakteryzujące przetwarzanie danych w Czasie Rzeczywistym [64] :

Określenie 2.

Mówimy, że przetwarzanie danych odbywa się w czasie rzeczywistym wtedy, gdy występują ograniczenia na czas obróbki nieodtwarzalnych danych dostarczonych z zewnątrz SC lub wtedy gdy wyniki przetwarzania oddziaływują w sposób bezpośredni i nieodwracalny na otoczenie SC.

Szczególną klasą SC powyższego typu są Komputerowe Systemy Sterujące zwane również Cyfrowymi Systemami Sterowania. Systemy te są bardzo silnie powiązane z otoczeniem, którym jest proces technologiczny. Rozpatrywać będziemy klasę KSS zwaną Systemami Sterowania Bezpośredniego /ang.: DDC - Direct Digital Control/.

Określenie 3.

Komputerowy System Sterujący /KSS/ rozumiany będzie jako taki system sterowania procesem technologicznym w czasie rzeczywistym, w którym algorytmy sterowania realizowane są przez uniwersalną maszynę cyfrową wydającą równocześnie sterowania na obiekt.

Oczywiście przy takim określeniu KSS obowiązują wszystkie pojęcia opisujące SC /zasób, zasób zużywalny, System Operacyjny itp/. Wszystkie dalsze rozważania dotyczące programów i SO w jednakowym stopniu dotyczą SC co i tak rozumianych KSS.

Rozszerzeniem sprzętu, a zarazem składnikiem SC umożliwiającym jego wykorzystanie, jest System Operacyjny.

W dalszych rozważaniach zwrócona będzie uwaga na rozdział zasobów, najważniejszą funkcję SO, mającą zasadniczy wpływ na efektywność pracy całego Systemu Cyfrowego. W związku z tym przyjęte zostanie następujące

Określenie 4

System Operacyjny /SO/ rozumiany będzie jako zbiór procedur rozdziału i udostępniania zasobów SC.

Pozostałe funkcje nie mają tak zasadniczego wpływu na efektywność SC, jednak w sposób pośredni wpływają na wykorzystanie zasobów SC [24] .

SO podejmuje wszystkie swoje działania w odpowiedzi na zdarzenia zachodzące w jego otoczeniu. Zdarzeniami wymagającymi reakcji SO jest np. żądanie lub zwolnienie zasobów przez którykolwiek z programów użytkowych. Wszędzie dalej zakłada się, że istnieje taki odcinek czasu, na którym żaden z programów nie zmienia swojego zapotrzebowania na zasoby. Oznaczymy długość tego odcinka jako τ. We wszystkich rozważaniach τ przyjmiemy jako podstawową jednostkę czasu.

Załóżmy, że SO jest aktywizowany cyklicznie, z okresem τ . Przyjmijmy również, że wszystkie czynności wykonywane przez SO trwają bardzo krótko i czas ten może być pominięty. Stanowi to pewne uproszczenie rzeczywistości i w dalszej części pracy zostaną przedstawione sytuacje, kiedy może być ono przyjęte.

Jedynymi, poza Systemem Operacyjnym, użytkownikami zasobów są programy wykonywane pod jego kontrolą. Wielkościami charakteryzującymi każdy z programów użytkowych są: czas obliczeń oznaczony przez M oraz żądania zasobów. Ponieważ zapotrzebowanie na zasoby na różnych etapach pracy programów może być różne, a każdy z programów żąda kilku rodzajów zasobów, do opisu zachowania programu w trakcie obliczeń użyjemy wektorowej funkcji czasu.

Niech $b^{(j)}(l)$ oznacza liczbę zasobów j-tego rodzaju niezbędną do kontynuacji pracy programu w l-tej chwili obliczeń, wówczas $b(l) = [b^{(1)}(l), b^{(2)}(l), \ldots, b^{r}(l)]^{T}$,

Zakładamy, że zarówno M jak i wektory $b(l)$, $l=1,2..,M$ są znane. Mogą one być bardzo łatwo wyznaczone podczas pracy programu bez przerwań.

Założenie o znajomości opisu każdego z programów jest do przyjęcia w przypadku komputerowych Systemów Sterujących, gdyż KSS składa się, ze skończonej liczby dobrze znanych i przetestowanych wcześniej programów.

W przypadku gdy program pracuje w systemie wieloprogramowym, a więc z przerwaniami, do określenia zapotrzebowania na zasoby w każdej chwili czasu potrzebna jest znajomości efektywnego czasu obliczeń programu.

Niech $\varkappa(k)$ oznacza czas efektywnego korzystania z zasobów /efektywny czas obliczeń/ do chwili k; wówczas żądanie zasobów w chwili k, oznaczane dalej przez $a(k)$ zdefiniowane jest w sposób następujący : $a(k) = b(\varkappa(k))$.

W przypadku, gdy w pewnej chwili program nie uzyska wszystkich wymaganych zasobów jego wykonywanie jest przerywane. Program pozostaje w tym stanie tak długo aż przydzielone zostaną mu potrzebne zasoby i jest kontynuowany od stanu w którym został przerwany. Po zakończeniu każdego taktu obliczeń o długości τ wszystkie zasoby wykorzystywane przez program są zwalniane, i są rozdzielane ponownie przez SO. Założenie to sprawia, że w systemie nie może dojść do sytuacji konfliktowych tzw. impasów /ang.: deadlock/ [65] .

SO w każdym takcie swojej pracy rozdziela zasoby. Jego decyzje mają charakter binarny tzn. określane są programy, które uzyskają wszystkie żądane zasoby oraz te, które nie otrzymają żadnego z żądanych zasobów - pozostaną w stanie zawieszenia. Decyzje SO dotyczące rozdziału zasobów oznaczone będą $v(k)$ i tak:

$v(k) = 1$ oznacza, że program w chwili k uzyska wszystkie żądane zasoby

$v(k) = 0$ oznacza, że program w chwili k nie będzie wykonywany.

Decyzje są podejmowane w ten sposób aby w żadnej chwili nie rozdysponować większej ilości zasobów niż jest dostępne w systemie.

Dla każdego programu w momencie jego zgłoszenia do systemu $\mathcal{X}(k)$ równa się 0. Dalsze zachowanie programu może być opisane w sposób następujący:

$$\mathcal{X}(k+1) = \mathcal{X}(k) + v(k) \qquad (1.1)$$

Jeżeli program w chwili k uzyska wszystkie żądane zasoby $(v(k)=1)$ będzie kontynuował obliczenia przez jeden takt - licznik czasu obliczeń zwiększy się o 1.

Jeżeli $v(k)=0$- program nie będzie wykonywał obliczeń i $\mathcal{X}/k+1/ = \mathcal{X}/k/$. Wszędzie dalej $\mathcal{X}(k)$ nazywane będzie stanem programu w chwili k. Łatwo zauważyć, że znajomość czasu obliczeń w chwili k oraz decyzji SO w tej chwili i następnych pozwala na dokładne opisanie zachowania programu.

U w a g a :

Powyżej podano formalny opis jednego programu w KSS. W związku z tym, że dalsze rozważania dotyczyć będą wielu programów wprowadzone wyżej oznaczenie a, b, ν, κ uzupełnione zostaną odpowiednimi indeksami pozwalającymi jednoznacznie określić numer programu, którego dotyczą.

Opis procesu technologicznego

W dalszej części tego rozdziału opisane będzie otoczenie KSS tzn. proces technologiczny.

Rozpatrywać będziemy typowe zadania [10] sterowania obiektami statycznymi i dynamicznymi.

Załóżmy, że rozpatrywany proces technologiczny składa się z I niezależnych obiektów sterowania nazywanych również podsystemami.
Dla każdego z obiektów określony jest wskaźnik jakości sterowania.

W przypadku gdy wszystkie podsystemy opisane są w sposób statyczny rozpatrywane będzie następujące zadanie sterowania dla każdego z obiektów:

dla każdego z_n^i znaleźć $u_n^i \in U_n^i$ mininalizujące kryterium jakości sterowania o postaci:

$$q^i = \sum_{n=0}^{N-1} \varphi_n^i(y_n^i) \qquad (1.2)$$

gdzie

$$y_n^i = f_n^i(u_n^i, z_n^i)$$

f_n^i - funkcja opisująca i-tybiekt sterowania w n-tym kroku sterowania,

φ_n^i - funkcja o wartościach rzeczywistych oceniająca jakość sterowania i-tym obiektem w n-tym kroku,

y_n^i - wyjście i-tego obiektu w n-tym kroku sterowania,

u_n^i - wielkość sterująca należąca do zbioru sterowań dopuszczalnych U_n^i,

z_n^i - zaburzenia oddziaływujące na i-tyobiekt, o zaburzeniach zakładamy, że mogą być mierzone,

N - liczba kroków sterowania,

y_n^i, u_n^i, z_n^i są wektorami należącymi do przestrzeni Euklidesowych o odpowiednich wymiarach. Wymiary te nie są określone ponieważ nie będą potrzebne w dalszych rozważaniach; f_n^i, φ_n^i są funkcjami odwzorowującymi przestrzenie Euklidesowe odpowiednich wymiarów.

Łatwo zauważyć, że rozwiązanie tak postawionego zadania sterowania prowadzi do następującej reguły wyznaczania sterowań optymalnych:

$$u^{*i}_n = \varphi^i_n(x^i_n), \quad i=1,2,\ldots,I; \quad n=0,1,\ldots,N-1 \tag{1.3}$$

W sytuacji gdy obiekty opisane są w sposób dynamiczny rozpatrywane będzie następujące zadanie sterowania optymalnego:

Znaleźć ciąg sterowań $u^i_n \in U^i_n$, $n=0,1,\ldots,N-1$ minimalizujących wskaźnik jakości sterowania o postaci:

$$Q^i = \sum_{n=0}^{N-1} \Phi^i_n(x^i_n, u^i_n) \tag{1.4}$$

gdzie:

$$x^i_{n+1} = F^i_n(x^i_n, u^i_n); \quad x^i_0 \text{ dane}, \; i=1,2,\ldots,I, \; n=0,1,\ldots,N-1$$

F^i_n - funkcja /wektorowa/ opisująca zmiany stanu i-tego obiektu

Φ^i_n - funkcja o wartościach rzeczywistych oceniająca jakość sterowania i-tym obiektem w n-tym kroku sterowania

x^i_n - stan i-tego obiektu w n-tym kroku sterowania

u^i_n - wymuszenie działające na obiekt należące do zbioru sterowań dopuszczalnych

x^i_0 - stan początkowy i-tego obiektu

N - liczba kroków sterowania; $N < \infty$

u^i_n, x^i_n są wektorami z przestrzeni Euklidesowych odpowiedniego wymiaru; F^i_n, Φ^i_n są funkcjami odwzorowywującymi przestrzenie Euklidesowe odpowiednich wymiarów. Ponieważ wymiary wszystkich przestrzeni nie będą istotne w dalszych rozważaniach, zostały tu pominięte.

Zakładać będziemy, że sterowania optymalne realizowane są w układzie ze sprzężeniem zwrotnym:

$$u^{*i}_n = \Psi^i_n(x^i_n) \tag{1.5}$$

Taki sposób realizacji sterowań ma szereg zalet. Główną jest to, że układy pracujące w układzie zamkniętym są mniej wrażliwe na niedokładności pomiaru stanu czy opisu obiektu.

Z drugiej strony sterowanie wg reguły /1.5/ ma bardzo pożyteczną własność, wynikającą z zasady optymalności Bellmana. Niezależnie od sterowania na pierwszym odcinku czasu $[0,\bar{m})$ sterowania $\gamma^i_m(x^i_m)$ dla $m \in [\bar{m},N)$ są optymalne /minimalizują cząstkowe kryterium $\bar{Q}^i$

$$\bar{Q}^i = \sum_{m=\bar{m}}^{N-1} \Phi^i_m(x^i_m, u^i_m) \quad /$$

UWAGA: Większość rozważań prowadzonych w dalszej części pracy może być łatwo uogólniona na następujące zadanie sterowania:

$$x^i_{m+1} = F^i_m(x^i_m, u^i_m, \alpha^i_m), \; x^i_0 \text{ dane} \qquad (1.6)$$

gdzie α^i_m jest ciągiem realizacji niezależnych zmiennych losowych o znanym rozkładzie prawdopodobieństwa.

Wskaźnik jakości sterowania ma postać:

$$\bar{Q}^i = \mathop{E}_{Z^i} \sum_{m=0}^{N-1} \Phi^i_m(x^i_m, u^i_m) \qquad (1.6')$$

Realizacja algorytmów sterowania w KSS

Algorytmy sterowania /1.3/ lub /1.5/ realizowane mogą być bądź przez urządzenie anologowe, bądź przez wyspecjalizowane urządzenie cyfrowe lub w sposób programowy z wykorzystaniem elektronicznej maszyny cyfrowej.
Jak już wskazywano, praca dotyczy tego ostatniego przypadku.

Praktycznie algorytm sterowania w KSS realizowany jest w następujący sposób [16] :

- procedury SO dokonują pomiaru pewnych wielkości z obiektu i umieszczają je w tablicach,
- uaktywniane są programy użytkowe, które pobierają z tablic dane, przetwarzają je wykorzystując zasoby SC, a wyniki umieszczają w pewnym wydzielonym obszarze pamięci,
- pod koniec kroku sterowania SO wyprowadza przetworzone dane na obiekt i dokonuje pomiarów.

Powyższy proces powtarzany jest cyklicznie. Jeżeli któryś z programów wyliczających sterowanie nie zakończy obliczeń w zadanym czasie, SO może postąpić na kilka sposobów: wydać na obiekt tą wartość, która już jest w tablicy /wyliczona w poprzednim takcie/, bądź umieścić w tablicy inne sterowanie przygotowane na wypadek zaistnienia takiej sytuacji.

Możliwych wariantów postępowania jest wiele. W każdej sytuacji spowoduje to pogorszenie wskaźnika jakości sterowanego procesu.

Opiszemy teraz kilka możliwych wariantów wyboru sterowań zastępczych:

- dla każdego z podsystemów określona jest stała w czasie, dopuszczalna i bezpieczna /tzn. taka, która nie powoduje zaburzeń w pracy sterowanego obiektu nawet wtedy gdy wydawane jest bardzo często/ wartość sterowania. W przypadku gdy nie istnieje stała w czasie dopuszczalna wartość tzn. $\bigcap_{m=0}^{N-1} U_m^i = \emptyset$, zakładamy istnienie ciągu dopuszczalnych sterowań zastępczych,
- wielkość wydawana na obiekt jest ekstrapolowana na podstawie sterowań wydanych dotychczas, najprostszym przypadkiem jest ekstrapolacja zerowego rzędu tzn: $\tilde{u}_m^i = u_{m-1}^i$
- określona jest bardzo prosta reguła wyliczenia sterowań zastępczych na podstawie znanych wartości zakłóceń /stanu/ obiektu.

Wybór sterowań zastępczych spełniających wyżej sprecyzowane wymagania musi być rozpatrywany indywidualnie w każdym konkretnym przypadku, w konsultacji z technologiem procesu. W całej pracy zakładać będziemy, że sposób określenia sterowań zastępczych jest dany.

Poniżej opisane zostaną podstawowe wielkości charakteryzujące każdy z programów użytkowych. Założymy, że w każdym kroku sterowania n sterowanie u_m^i wyliczone według zależności (1.3) lub (1.5) przez osobny program. Przyjmujemy następujące oznaczenia:

p_n^i - program wyliczający u_m^i wg zależności (1.3) lub (1.5),

T^i - długość kroku sterowania i-tym obiektem,

M_n^i - czas obliczeń programu p_n^i, $M_m^i \leq T^i$

r_n^i - moment przyjścia programu p_n^i do systemu, $r_m^i = n \cdot T^i$

d_n^i - chwila do której program p_n^i powinien zakończyć obliczenia, $d_m^i = (n+1)T^i$

$x_m^i(k)$ - stan programu p_n^i w chwili k, $x_m^i(k) = 0$, $k \leq r_m^i$

$a_m^i(k)$ - zapotrzebowanie na zasoby programu p_n^i w chwili k,

UWAGA: Wszędzie w pracy przyjęto następujące znaczenie nierówności pomiędzy wektorami I wymiarowymi :

$x \in R^I$, $y \in R^I$

$x \leqslant y \quad \forall i \quad x^i \leqslant y^i$

$x > y \quad \exists i \quad x^i > y^i$

1.3. Formalne postawienie problemu pracy

Niżej postawione zostanie, z wykorzystaniem wprowadzonych oznaczeń, zadanie optymalnego rozdziału zasobów Komputerowego Systemu Sterującego. Zasoby te powinny być rozdzielone w ten sposób aby efektyność KSS była jak największa.

Efektywność Komputerowego Systemu Sterującego

Jak wiadomo podstawową funkcją SO jest taki rozdział zasobów aby efektywność SC była jak największa. Powstaje problem jak tą efektywność mierzyć. Jest ona zawsze zależna od środowiska w którym pracuje SC. W dużym ośrodku obliczeniowym najważniejsze jest takie planowanie zadań aby wykorzystanie zasobów SC było jak największe. W przypadku systemu interekcyjnego podstawowym miernikiem efektywności jest czas reakcji [72]. W innych sytuacjach zależeć nam może na minimalizacji średnich długości kolejek - zajmują one bowiem pamięć SC.

Oczywiście każde z powyższych kryteriów może być stosowane do wyboru algorytmów SO KSS [29,30,45,46,50,51], wszystkie z nich mają jednak pewną wadę: mają niewielki i jedynie pośredni związek z głównym celem KSS jakim jest sterowanie procesem technologicznym.

Pożądaną własnością KSS jest zdolność wykonania wszystkich zadań przed ich liniami krytycznymi [4,5,9,49].

W przypadku, gdy nie można tego zapewnić, najbardziej naturalnym, wydaje się jednak przyjęcie jako miary efektywności, wskaźnika jakości sterowanego procesu technologicznego [33,55].

W przypadku rozpatrywanym w pracy jako miarę efektywności pracy KSS przyjęta będzie pewna funkcja zależna od wskaź-

ników jakości wszystkich obiektów Q^i, i=1,2,...,I

$$Q = \bar{\Phi}(Q^1, Q^2, \ldots, Q^I) \tag{1.7}$$

np:
$$Q = \sum_{i=1}^{I} a^i Q^i$$

Funkcja $\bar{\Phi}$ powinna być funkcją niemalejącą względem każdego, że swych argumentów, przy ustalonych wartościach pozostałych zmiennych niezależnych.

Tak rozumiana efektywność zależy od sterowań wydanych na proces technologiczny przez programy uruchamiane pod kontrolą SO. Sterowania wydawane przez KSS zależą od wykonania, bądź niewykonania programów je realizujących. To zaś zależy od sposobu rozdziału zasobów przez SO. Tak więc zasadniczy wpływ na efektywność KSS ma jego System Operacyjny.

Przedstawimy teraz podstawowe zadania optymalizacji rozdziału zasobów badane w pracy.

ZADANIE P1. ROZDZIAŁ ZASOBÓW W KSS OBIEKTAMI STATYCZNYMI

Znaleźć ciąg liczb $v_n^{*i}(k)$, $nT \leq k < (n+1)T$, n=0,1,...,N-1
i = 1,2,...I minimalizujący wskaźnik jakości Q^I

$$Q^I = \sum_{i=1}^{I} a^i \sum_{n=0}^{N-1} \varphi_n^i\left(f_n^i\left(\gamma_n^i\left(x_n^i((n+1)T), z_n^i\right), z_n^i\right)\right)$$

gdzie

$$x_n^i(k+1) = x_n^i(k) + v_n^i(k)$$

$$x_n^i(k) = 0, \quad k \leq nT$$

przy ograniczeniach

$$\forall k \sum_{i=1}^{I} q_n^i(k) \cdot v_n^i(k) \leq C, \quad n = \left[\frac{k}{T}\right], \quad v_n^i(k) \text{ binarne}$$

$$f_n^i(x, z_n^i) = \begin{cases} \gamma_n^i(z_n^i) & x = M_n^i \\ \tilde{u}_n^i & x < M_n^i \end{cases} \tag{1.8}$$

$\tilde{u}_n^i$ - wartość sterowania zastępczego wydawana na i-ty obiekt w n-tym kroku sterowania,

$z_n^i, \varphi_n^i, f_n^i$ - zostały opisane podczas omawiania zadania sterowania obiektami statycznymi,

$x_n^i(k)$ - stan i-tego programu w n-tym kroku sterowania w chwili k,

$a^i_n(k), v^i_n(k), c$ - omówione zostały podczas definiowania pojęcia zasobu i programu.

Jako szczególny przypadek zadania P1 rozpatrywane będzie zadanie P1' różniące się tym, że rozdzielany będzie tylko jeden zasób tzn. $\dim(c) = r = 1$. Jest to bardzo ważne z praktycznego punktu widzenia zadanie szeregowania programów na wielu procesorach.

ZADANIE P2 ROZDZIAŁ ZASOBÓW W KSS OBIEKTAMI DYNAMICZNYMI

Znaleźć ciąg liczb $v^{i*}_n(k)$, $nT \le k < (n+1)T$, $n = 0,1,..,N-1$

$i = 1,2,...I$ minimalizujących wskaźnik jakości Q^{II}

$$Q^{II} = \sum_{i=1}^{I} a^i \sum_{n=0}^{N-1} \Phi^i_n\left(x^i_n, \gamma^i_n\left(x^i_n((n+1)T), x^i_n\right)\right)$$

gdzie

$$x^i_{n+1} = F^i_n\left(x^i_n, \gamma^i_n\left(x^i_n((n+1)T), x^i_n\right)\right)$$

$$x^i_n(k+1) = x^i_n(k) + v^i_n(k)$$

$$x^i_n(k) = 0, \quad k \le nT$$

$$i = 1,2,..,I, \quad n = 0,1,...,N-1$$

przy ograniczeniach

$$\forall_k \sum_{i=1}^{I} a^i_n(k) \cdot v^i_n(k) \le c, \quad v^i_n(k) \text{ binarne}, \quad n = \left[\frac{k}{T}\right]$$

$$\gamma^i_n(x, x^i_n) = \begin{cases} \Psi^i_n(x^i_n) & x = M^i_n \\ \tilde{u}^i_n & x < M^i_n \end{cases} \qquad (1.9)$$

x^i_n, F^i_n, Φ^i_n - omówiono przy okazji zadania sterowania obiektami dynamicznymi

znaczenie $\tilde{u}^i_n, x^i_n(k), a^i_n(k), v^i_n(k), c$ jest takie same jak w zadaniu P1.

UWAGA: użyto tego samego oznaczenia γ^i_n na funkcję określającą sterowania wydawane na obiekt statyczny i dynamiczny w celu podkreślenia podobieństw pomiędzy obu zadaniami.
Ponieważ oba zadania nie będą rozpatrywane równocześnie nie doprowadzi to do nieporozumień.

Analogicznie jak P1' rozpatrywane będzie zadanie P2' rozdziału jednego zasobu /np.: procesora/. Zadanie to ma ogromne znaczenie praktyczne. Z takim problemem mamy do czynienia najczęściej podczas projektowania KSS.

Jak widać są to skomplikowane problemy optymalizacji /sterowania/ o binarnych zmiennych decyzyjnych. Ogólnie postawione zadanie jest problemem o nieliniowej funkcji celu i nieliniowych ograniczeniach, ponieważ

$$a_n^i(k) = b_m^i\left(x_m^i(k)\right).$$

Problem pracy sformułowany być może w sposób następujący:

- wyznaczyć algorytmy rozwiązania zadań P1, P1' , P2 i P2',
- rozważyć możliwość stosowania optymalnych algorytmów rozdziału zasobów w Systemach Operacyjnych Komputerowych Systemów Sterujących,
- porównać wyniki osiągnięte przy stosowaniu optymalnego rozdziału z wynikami możliwymi do uzyskania przy zastosowaniu klasycznych algorytmów rozdziału zasobów.

2. ROZDZIAŁ ZASOBÓW W KOMPUTEROWYM SYSTEMIE STERUJĄCYM OBIEKTAMI STATYCZNYMI.

W rozdziale tym omówione zostanie zadanie rozdziału zasobów w Komputerowym Systemie Sterującym obiektami statycznymi w przypadku, gdy długość kroku sterowania dla każdego podsystemu jest jednakowa.

2.1. Rozdział zasobu limitującego wykorzystanie Systemu Cyfrowego.

2.1.1. Własności optymalnego rozdziału zasobów.

W rozdziale tym rozpatrzony zostanie problem rozdziału jednego zasobu /P1'/, który po prostych przekształceniach może być zapisany w następującej postaci:

znaleźć binarne liczby $u^{*i}_m(k)$ miminalizujące funkcję Q'^I

$$Q'^I = \sum_{i=1}^{I} \alpha^i \sum_{m=0}^{N-1} \varphi^i_m\left(f^i_m\left(\gamma^i_m\left(\sum_{k=mT}^{(m+1)T-1} v^i_m(k), z^i_m\right), z^i_m\right)\right) \qquad (2.1)$$

przy ograniczeniach

$$\forall_k \sum_{i=1}^{I} v^i_m(k) \leq c \quad , \quad m = \left[\frac{k}{T}\right] , \quad \dim c = 1 \qquad (2.2)$$

gdzie φ^i_m , f^i_m są określone wzorem /1.2/ a γ^i_m wzorem /1.8/.

Powyższy problem - rozdziału jednego zasobu ma ogromne znaczenie praktyczne, jest też najszerzej rozpatrywany w literaturze. Może on dotyczyć sytuacji rozdziału procesora KSS - mamy wówczas do czynienia z problemem szeregowania zadań na c procesorach. Obejmuje on również sytuację, gdzie jeden z zasobów występuje w niedoborze i wykorzystywany jest stale przez wszystkie programy. Wówczas taki zasób ogranicza wykorzystanie innych zasobów w SC. Sytuacja taka zachodzić może np. w wieloprocesowym systemie pracującym "na wspólną pamięć" lub w systemie, w którym wszystkie programy do swej pracy wymagają pewnego urządzenia we/wy lub określonego zbioru danych.

Wszędzie dalej zakładać będziemy, że chodzi o rozdział precesora, i że w każdej chwili program może żądać jedynie jednego procesora, tzn. $a^i_n = 1$.

Poniżej opisane zostaną podstawowe własności rozwiązywanego problemu:

1° Zadanie /2.1./, /2.2./ jest równoważne N zadaniom optymalizacji.

Dla każdego $n = 1,2...N$ wybrać tak liczby $v^{*i}_n(k)$, $i=1,2,...,I$, $nT \le k < (n+1)T$

aby minimalizować $\bar{Q}_n$

$$\bar{Q}_n = \sum_{i=1}^{I} \alpha^i \varphi^i_n\left(f^i_n\left(\gamma^i_n\left(\sum_{k=nT}^{(n+1)T-1} v^i_n(k), z^i_n\right), z^i_n\right)\right) \qquad (2.3)$$

przy ograniczeniach:

$$\forall k \quad \sum_{i=1}^{I} v^i_n(k) \le c \;, \quad v^i_n(k) \text{ binarne} \qquad (2.4)$$

Jest to oczywiste, ze względu na sumacyjną postać kryterium i rozłączność zbiorów ograniczeń w poszczególnych krokach sterowania.

2° Wartość kryterium Q^I nie zależy od kolejności wykonania programów, a jedynie od faktu ich wykonania.

Własność ta wynika bezpośrednio ze sformułowania problemu /por 1.3./ i jest własnością ogólną wszystkich zadań rozpatrywanych w tej pracy.
Ważną jej konsekwencją jest to, że SO będzie przydzielał zasoby jedynie tym programom, które zostaną wykonane w całości.

Korzystając z powyższych dwu własności dokonamy przekształcenia każdego z zadań /2.3./ do prostszej postaci.

Sumując stronami ograniczenia /2.4./ oraz zmieniając kolejność sum otrzymamy:

$$\sum_{i=1}^{I} \sum_{k=nT}^{(n+1)T-1} v^i_n(k) \le cT$$

Oznaczmy $\bar{w}^i_n = \sum_{k=nT}^{(n+1)T-1} v^i_n(k)$

Oczywiście $\bar{w}^i_n \le M^i_n$, a korzystając z własności 2°

wiemy, że w rozwiązaniu optymalnym $\bar{w}_n^i$ przyjmować będzie jedynie dwie wartości 0 i M_n^i; wprowadzając nowe oznaczenia:

$$w_n^i = \begin{cases} 0 \text{ jeżeli } \bar{w}_n^i < M_n^i \\ 1 \text{ jeżeli } \bar{w}_n^i = M_n^i \end{cases} \qquad i=1,2,..,I;\ n=0,1,..,N-1$$

ograniczenia możemy zapisać w postaci:

$$\sum_{i=1}^{I} M_n^i w_n^i \leq cT \quad , \quad n=0,1,..,N-1 \tag{2.5}$$

Łatwo zauważyć, że dla każdego układu liczb $v_n^i(k)$ będących rozwiązaniem optymalnym zadania /2.3./ /2.4/ istnieją liczby w_n^i binarne spełniające nierówność /2.5./

Pokażemy teraz, w sposób konstrukcyjny, że dla dowolnych w_n^i spełniających /2.5./ istnieją $v_n^i(k)$ spełniające /2.4./

Weźmy dowolny układ liczb $v_n^i(k)$ takich, że:

$$\sum_{k=nT}^{(n+1)T-1} v_n^i(k) = M_n^i \cdot w_n^i$$

/np.: $\forall i$ $v_n^i(k) = w_n^i$, $k=nT, nT+1, .., nT+M_n^i-1$, i $v_n^i(k) = 0$ dla pozostałych k/

Jeżeli $\forall$ k $\sum_{i=1}^{I} v_n^i(k) \leq c$ to liczby $v_n^i(k)$, i=1,2,...,I $nT \leq k < (n+1)T$ spełniają ograniczenia /2.4/

W przeciwnym razie istnieje takie $\hat{k}$, że:

$$\sum_{i=1}^{I} v_n^i(\hat{k}) = c' > c$$

oraz

$$\exists\ \bar{k} \quad \sum_{i=1}^{I} v_n^i(\bar{k}) < c$$

/w przeciwnym bowiem razie, jeżeli $\forall$ k $\sum_{i=1}^{I} v_n^i(k) \geq c$ to

$$\sum_{k=nT}^{(n+1)T-1} \sum_{i=1}^{I} v_n^i(k) \geq cT$$ co przeczy założeniu, że ograniczenia /2.5/ są spełnione/.

Wybierzmy $\bar{i}$ tak, że $v_n^{\bar{i}}(\hat{k}) = 1$ i $v_n^{\bar{i}}(\bar{k}) = 0$

/takie $\bar{i}$ istnieje, ponieważ $\sum_{i=1}^{I} v_n^i(\hat{k}) > \sum_{i=1}^{I} v_n^i(\bar{k})$ /

Dokonajmy następującej zamiany

$$v_m^{\bar{i}}(\hat{k}) = 1 \quad , \quad v_m^{\bar{i}}(\bar{k}) = 1,$$

nie spowoduje ona naruszenia ograniczeń w chwili $\bar{k}$

natomiast w chwili: $\hat{k}$ $\sum_{i=1}^{I} v_m^i(\hat{k}) = c'' < c'$

Powyższy proces powtarzamy tak długo, aż wszystkie ograniczenia, będą spełnione. Łatwo zauważyć, że wykonać to należy skończoną liczbą razy; w każdej iteracji nie powodujemy naruszenia żadnego ograniczenia, zmniejszając stopień niespełnienia /tzn. $c'' < c$ / jednego z ograniczeń.

Pokazaliśmy, że ograniczenia /2.4/ z dodatkowym ograniczeniem

$$\forall i \quad \sum_{k=mT}^{(m+1)T-1} v_m^i(k) = \begin{cases} 0 \\ M_m^i \end{cases} \text{lub}$$

są równoważne ograniczeniu /2.5/. Z własności 2^0 wynika, że dla każdego rozwiązania optymalnego zadania /2.3.//2.4/ istnieje równoważne mu /w sensie kryterium/ rozwiązanie spełniające powyższe, dodatkowe ograniczenie. Tak więc zamiast minimalizować funkcję /2.3/ wystarczy minimalizować następującą funkcję

$$Q_m = \sum_{i=1}^{I} \alpha^i \varphi_m^i\left(f_m^i\left(\gamma_m^i\left(w_m^i \cdot M_m^i, z_m^i\right), \bar{z}_m^i\right)\right) \qquad (2.6)$$

przy ograniczeniach /2.5/.

Zauważmy, że /2.6/ można dalej przekształcić:

$$Q_m = \sum_{i=1}^{I} \alpha^i \varphi_m^i\left(f_m^i\left(w_m^i \cdot \psi_m^i(z_m^i) + (1-w_m^i)\tilde{u}_m^i, z_m^i\right)\right) =$$

$$= \sum_{i=1}^{I} \alpha^i \left[w_m^i \varphi_m^i\left(f_m^i\left(\psi_m^i(z_m^i), z_m^i\right)\right) + (1-w_m^i)\varphi_m^i\left(f_m^i(\tilde{u}_m^i, z_m^i)\right)\right] =$$

$$= \sum_{i=1}^{I} \alpha^i w_m^i \left[\varphi_m^i\left(f_m^i\left(\psi_m^i(z_m^i), z_m^i\right)\right) - \varphi_m^i\left(f_m^i(\tilde{u}_m^i, z_m^i)\right)\right] + \alpha^i \varphi_m^i\left(f_m^i(\tilde{u}_m^i, z_m^i)\right)$$

oznaczmy:

$$\delta_m^i(z_m^i) = \alpha^i \left[\varphi_m^i\left(f_m^i\left(\psi_m^i(z_m^i), z_m^i\right)\right) - \varphi_m^i\left(f_m^i(\tilde{u}_m^i, z_m^i)\right)\right] \qquad (2.7)$$

W powyższym wzorze składnik $\sum_{i=1}^{I} \alpha^i \varphi_m^i\left(f_m^i(\tilde{u}_m^i, z_m^i)\right)$ nie ma wpływu na wybór optymalnych w_m^i, a jedynie na wartość wskaźnika jakości i może być pominięty podczas wyliczania w_m^{*i}.

3° Jak pokazano wyżej zadanie /2.3./, /2,4/ równoważne jest dwu poniższym zadaniom

a/ Dla n = 0,1,...N-1 znaleźć $\overset{*i}{w}_n$, $i=1,2,...I$ minimalizujące dla każdego $\overset{i}{z}_n$ funkcję Qn

$$Q_n = \sum_{i=1}^{I} w_n^i \cdot \delta_n^i(z_n^i) \qquad (2.8)$$

przy ograniczeniach:

$$\sum_{i=1}^{I} m_n^i \cdot w_n^i \leq cT \ , \ w_n^i \text{ binarne} \qquad (2.9)$$

b/ Na podstawie $\overset{*i}{w}_n$ znaleźć liczby $\overset{*i}{\delta}_n(k)$, $i=1,2,...,I$; $nT \leq k < (n+1)T$ spełniające ograniczenia /2.4/.

Zadanie a/ dla każdego $\overset{i}{z}_n$, $i=1,2,...I$ jest typowym zadaniem załadunku o zmiennych binarnych, natomiast zadanie b/, jak łatwo zauważyć jest zadaniem szeregowania programów przed linią krytyczną na c procesorach.

Udowodniona własność 3° prowadzi w sposób naturalny do dekompozycji procedur szeregowania SO.
W znanych pracach teoretycznych i SO stosowano takie podejście nie zawsze je uzasadniając.

4° W przypadku gdy c=1, tzn. w SC jest tylko jeden procesor, zadanie b/ jest bardzo proste i jak pokazano w [23] programy mogą być wykonywane bez przerwań. Oznacza to, że wszystkie programy dla których $w_n^i = 1$ mogą być wykonane w dowolnej kolejności.

W dalszej części rozdziału dokonamy krótkiego porównania otrzymanych wyników z metodami szeregowania stosowanymi dotychczas. Ponieważ najprostszym i najpowszechniejszym sposobem szeregowania programów jest szeregowanie z uwzględnieniem priorytetów, spróbujemy opisać uzyskane wyniki w języku priorytetów.
W naszym przypadku reguła wyznaczania priorytetów jest stała /zadanie a//, ponieważ parametry w równaniu /2.8/ zależą od zaburzeń zewnętrznych, wyznaczone priorytety są zmienne w czasie. W związku z tym wszystkie reguły z priorytetami stałymi

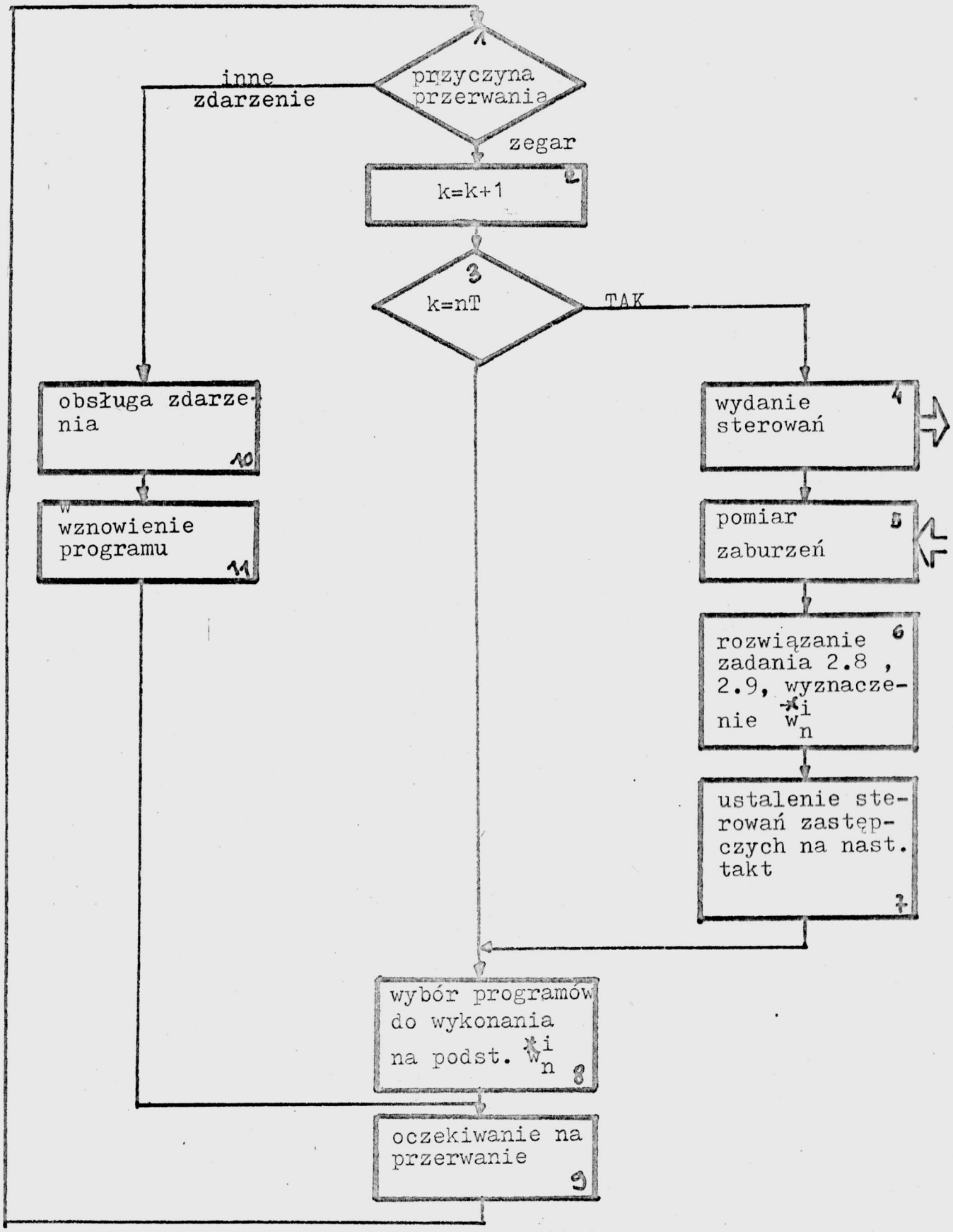
w czasie będą na ogół nieoptymalne. W przypadku, gdy obiekt sterowania charakteryzuje się następującą własnością: pewne podsystemy mają niewielki wpływ na kryterium tzn. oddziaływujące zaburzenia są niewielkie, a sterowanie zastępcze bliskie optymalnemu; inne natomiast poddane są silnym zaburzeniom; priorytety stałe w czasie mogą okazać się bliskie optymalnym.

Zauważmy również, że jeżeli znamy pewne własności oddziaływujących na obiekt zaburzeń np.: ich trendy, zakres zmienności itp. można budować reguły z priorytetami zmiennymi w czasie bliskie optymalnym.

Omówimy teraz pokrótce sposób realizacji algorytmów optymalnego rozdziału zasobów w SO KSS.
Na rysunku 2.1. przedstawiono bardzo uproszczony schemat blokowy procedury szeregowania SO. Strzałki z prawej strony bloków o numerach 4 i 5 symbolizują związek SO z obiektem; k oznacza licznik czasu.

Zauważmy, że zadania a/ i b/ rozwiązane być muszą jednokrotnie w każdym kroku sterowania.

Dalsze uwagi na temat realizacji algorytmów optymalnego rozdziału zasobów podane będą w rozdziale 5.



Rys. 2.1. Uproszczony schemat blokowy procedury szeregowania SO.

2.1.2. Opis algorytmu rozdziału zasobów.

Podane zostaną tutaj ogólne uwagi dotyczące rozwiązywania zadań postawionych w 2.1.1.

Do rozwiązywania zadania /2.8/ stosować można dowolny algorytm programowania całkowitoliczbowego liniowego. W szczególności, ze względu że jest tylko jedno ograniczenie można stosować metody rozwiązywania zadań załadunku o ograniczonych zmiennych decyzyjnych.

Wykorzystując pewne własności funkcji celu oraz ograniczeń można przyśpieszyć działanie procedur optymalizacji zastępując ograniczenia /2.9/ silniejszym [42] lub próbując ustalić wartości pewnych zmiennych wcześniej [37] .

W każdym jednak przypadku algorytmy rozwiązywania załadunku są dosyć efektywne, mimo iż sam problem jest NP-zupełny [2] .

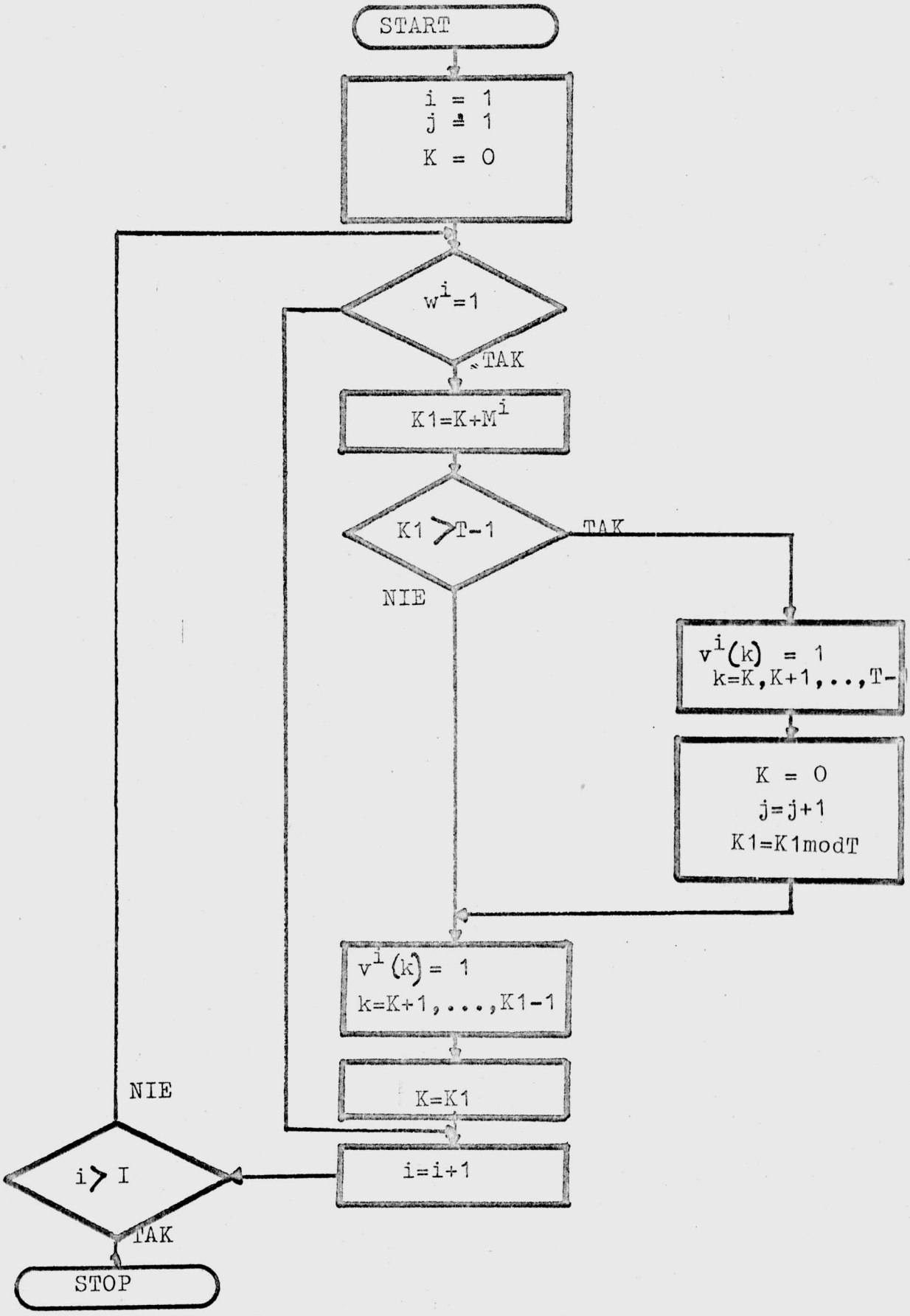
Poniżej podamy metody rozwiązywania zadania b/ z 2.1.1.

Do znalezienia uszeregowania dopuszczalnego można stosować algorytm zaproponowany w 2.1.1., który jest bardzo prosty, ale niezbyt efektywny. W dalszej części rozdziału podany zostanie jego schemat blokowy.

Z drugiej jednak strony jest to zadanie szeregowania przed liniami krytycznymi na c jednakowych procesorach. Problem ten posiada bogatą literaturę, np. [4,5,6,9,20,49] .

Szczególnie wiele rezultatów otrzymano dla zadań szeregowania na jednym lub dwu procesorach.

Jest oczywiste, że w sytuacji kiedy mamy pewność, że dla wybranych programów uszeregowanie przed liniami krytycznymi istnieje, oraz dla wszystkich programów jest określona jednakowa linia krytyczna wówczas można użyć również algorytmów szeregowania gdzie kryterium jest minimalizacja czasu wykonania zadań. Bardzo efektywny algorytm rozwiązania tego zadania podano w [4] . Na rysunku 2.2. podany jest schemat blokowy tego algorytmu zapisany w notacji obowiązującej w tej pracy.



Rys. 2.2. Schemat blokowy procedury rozdziału zasobów.

oznaczenia:

i - nr programu,

j - nr procesora,

M^i - czas obliczeń programu i,

k - nr taktu rozdziału zasobów,

T - długość odcinka, na którym szeregowane są programy /długość kroku sterowania/.

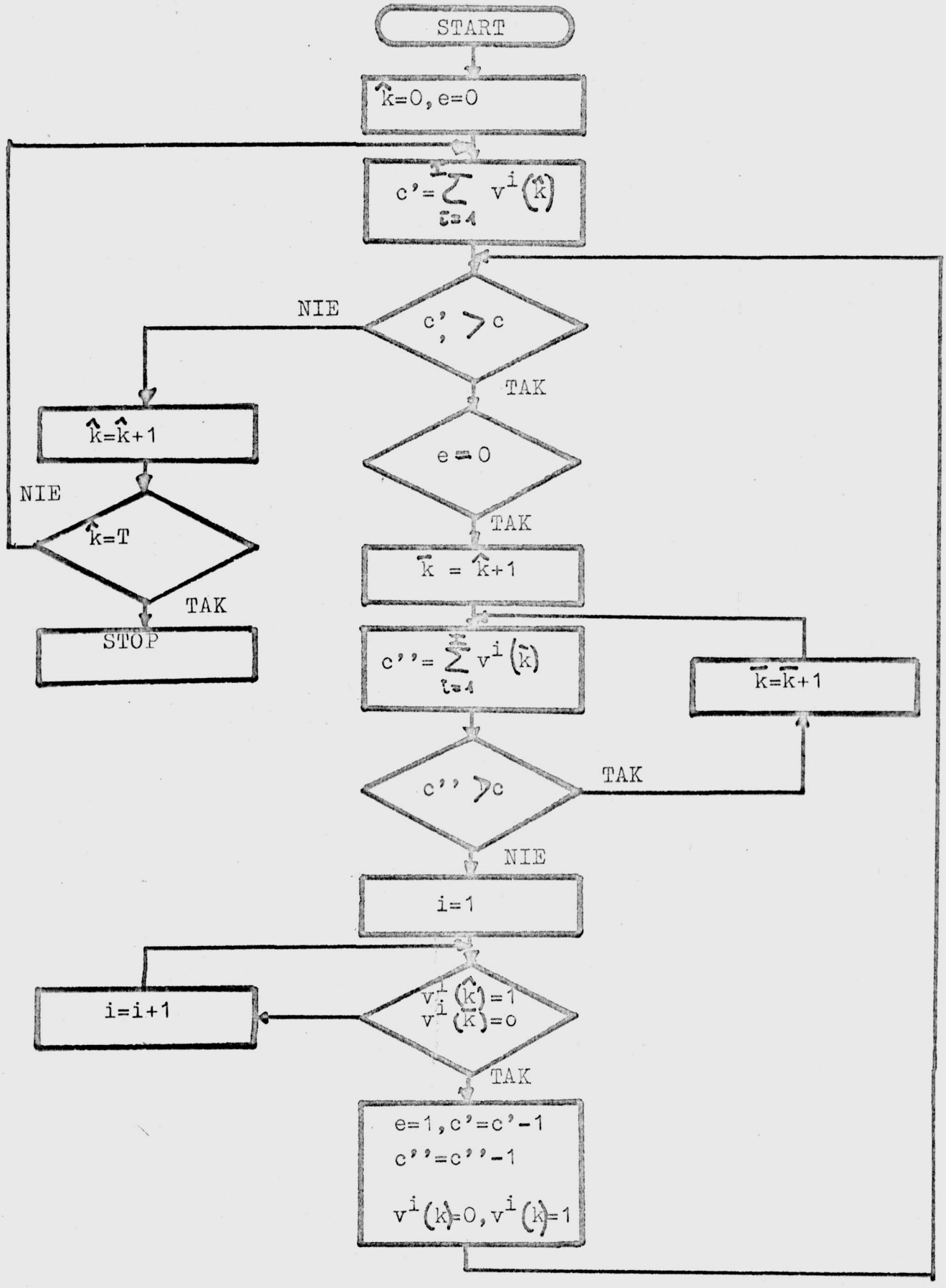
Na schemacie blokowym, w sposób symboliczny zaznaczono przydział programu do procesora.

Ideę algorytmu można przestawić w następujący sposób:

1. Rozpocznij wykonywanie pierwszego programu, dla którego $w_n^i = 1$, w chwili $k = 0$ na procesorze 1.

2. Wybierz następny program dla którego $w_n^i = 1$ i rozpocznij jego wykonanie na tym samym procesorze w chwili zakończenia wykonywania poprzedniego programu.
Powtarzaj ten krok do chwili gdy wszystkie zadania zostaną uszeregowane lub $k = T$.

3. Część zadania pozostająca do wykonania po osiągnięciu $k = T$ przydziel do następnego procesora rozpoczynając jego wykonanie od chwili k = 0. Wróć do kroku 2.



Rys. 2.3. Schemat blokowy procedury szeregowania programów opisanej w 2.1.1.

Na rysunku 2.3 przestawiony jest schemat algorytmu podanego w 2.1.1. Konstruowany on był dla potrzeb dowodu i w związku z tym jest na ogół mniej efektywny niż algorytm poprzedni.
W pewnych jednak warunkach może mieć zalety. Rozpoczyna on pracę od dowolnego uszeregowania, w szczególności uszeregowaniem początkowym może być uszeregowanie programów z poprzedniego kroku, z którego odrzucano programy niewykonywane a dołożono programy, które powinny być wykonane. W tej sytuacji algorytm może pracować krócej. Inną jego zaletą jest to, że może on być stosowany z taktu na takt, gdy algorytm poprzedni musi być wykonany w całości przed rozpoczęciem wykonywania programów.

Przykład

Rozpatrzmy proces technologiczny złożony z I obiektów technologicznych, każdy opisany równaniem /1.2./.
Niech każdy z obiektów ma następującą własność: minimalna wartość wskaźnika jakości w każdym kroku sterowania, tzn.
$\varphi_n^i(t_n^i(\psi_n^i(z_n^i), z_n^i)) = \bar{\varphi}^i$ jest stała i nie zależy ani od zakłóceń, ani od czasu.
Oznacza to, że zakłócenia zmieniają jedynie punkt pracy, nie zmieniając np. granicznej do uzyskania wydajności lub kosztu.

Wskaźnik jakości sterowania ma postać:

$$Q = \sum_{i=1}^{I} \alpha^i \sum_{n=0}^{N-1} \varphi_n^i(t_n^i(u_n^i, z_n^i))$$

Przyjmijmy, że SC składa się jedynie z jednego procesora tzn. c=1. Załóżmy, że dla każdego $n = 0,1,\ldots,N-1$

$$M_n^1 = M_n^2 = \ldots = M_n^I = M_n$$

(czasy pracy programów są jednakowe).
Niech $m_n = \lfloor T/M_n \rfloor$ oznacza maksymalną liczbę programów jaka może być wykonana w n-tym kroku sterowania, $m_n < I$.

Jak wiadomo zadanie wyjściowe może być rozbite na N zadań:

$$\forall n \forall z_n^i \quad \min_{w_n^i} \sum_{i=1}^{I} w_n^i \delta_n^i(z_n^i)$$

$$i = 1,2,\ldots I$$

przy ograniczeniach:

$$\sum_{i=1}^{I} w_n^i \le m_n$$

Dla danego z_n^i uporządkujmy liczby $\delta_n^i(z_n^i)$ w kolejności rosnącej tzn.:

$$\delta_n^{i_1}(z_n^{i_1}) \le \delta^{i_2}(z_n^{i_2}) \le \dots \le \delta^{i_I}(z_n^{i_I})$$

Łatwo zauważyć, że rozwiązanie optymalne ma postać:

$$w_n^{i_1} = 1, \; w_n^{i_2} = 1, \dots, w_n^{i_{m_n}} = 1, w_n^{i_{m_n+1}} = 0, \dots, w_n^{i_I} = 0.$$

Realizująca taki algorytm rozdziału zasobów procedura SO ma postać:

dla każdego i wylicza na podstawie zmierzonych zaburzeń

$$\delta_n^i(z_n^i) = \alpha^i [\bar{\varphi}^i - \varphi_n^i(f_n^i(\tilde{u}_n^i, z_n^i))]$$

i traktuje liczbę $\delta_n^i(z_n^i)$ jako priorytet programu i /przyjmujemy konwencję im mniejsza wartość priorytetu tym większa "ważność" programu/. □

2.2. Rozdział wielu zasobów

2.2.1. Własności zadania rozdziału wielu zasobów.

Zadanie rozdziału wielu zasobów w KSS obiektami statycznymi /P1/ może być zapisane w następujący sposób /por. 2.1.1./;

znaleźć liczby binarne $v_n^{*i}(k)$, $i=1,2,..,I; n=0,1,...,N-1$ minimalizujące Q^I

$$Q^I = \sum_{i=1}^{I} \alpha^i \sum_{n=0}^{N-1} \varphi_n^i(f_n^i(\gamma_n^i(\sum_{k=nT}^{(n+1)T-1} v_n^i(k), z_n^i), z_n^i)) \qquad (2.10)$$

przy ograniczeniach

$$\forall_k \sum_{i=1}^{I} v_n^i(k) \cdot a_n^i(k) \le c \quad , n = [\tfrac{k}{T}] \qquad (2.11)$$

Podobnie jak w 2.1.1. problem może być zapisany jako N niezależnych zadań o postaci:

dla każdego $0 \leq n < N$ znaleźć liczby $v_n^{*i}(k)$, $i=1,2,..,I$; $nT \leq k < (n+1)T$

$$Q_m^I = \sum_{i=1}^{I} \alpha^i \varphi_m^i \left(f_m^i \left(b_m^i \left(\sum_{k=mT}^{(m+1)T-1} v_m^i(k), a_m^i \right), z_m^i \right) \right) \quad \text{minimalizujące} \qquad (2.12)$$

przy ograniczeniach

$$\sum_{i=1}^{I} v_m^i(k) \cdot a_m^i(k) \leq c$$

Zadanie to jest nieliniowym zadaniem programowania binarnego. Ograniczenia mają również nieliniową postać, ponieważ /por. rozdz. 1.2./

$$a_m^i(k) = b_m^i \left(\sum_{j=mT}^{k} v_m^i(j) \right)$$

Funkcję celu można sprowadzić do postaci liniowej /2.8./ poprzez wprowadzenie nowych zmiennych decyzyjnych /podobnie jak pokazano to w 2.2.1./ Ograniczenia pozostają jednak ciągle nieliniowe.

Do rozwiązania zadania /2.12/ można stosować dowolne metody programowania binarnego /np. metodę podziału i ograniczeń/. Jednak postać zadania nie pozwala przypuszczać, że otrzymane algorytmy będą efektywne.

W rozdziale 2.2.2. podana zostanie metoda konstruowania zadania równoważnego problemom 2.1.2/. Będzie to zadanie liniowe o większej liczbie zmiennych i ograniczeń ale tym samym zbiorze rozwiązań dopuszczalnych i funkcji celu równoważnej /2.8./.

2.2.2. Algorytm optymalnego rozdziału wielu zasobów.

Poniżej zaproponowany zostanie algorytm sprowadzania każdego z N zadań /2.12/ do zadania szukania całkowitoliczbowego przepływu o minimalnym koszcie w pewnej sieci z mnożnikami.

Sieć z mnożnikami określona jest w sposób następujący [8, 18, 19, 39, 66, 67, 68]:

Niech $G = \{\mathcal{W}, \mathcal{A}\}$ będzie grafem skierowanym o wierzchołkach $s, p_1, p_2, \ldots, p_l, t \in \mathcal{W}$ i łukach $e_1, e_2, \ldots, e_g \in \mathcal{A}$

Niech Γp oznacza zbiór następników wierzchołka p /w sensie relacji wyznaczonej przez łuki grafu/, a $\Gamma^{-1} p$, zbiór poprzedników p [35].

W grafie G spełniane są następujące warunki:

- z każdym wierzchołkiem sieci p związana jest liczba rzeczywista η_p zwana mnożnikiem,
- z każdym łukiem e /określonym przez parę wierzchołków / związana jest liczba E_e zwana jego przepustowością, oraz liczba λ_e zwana jednostkowym kosztem przepływu /jeżeli $\lambda_e < 0$ nie ma interpretacji kosztu przepływu, jednak nazwa ta zostanie utrzymana/.

Niech ε_e oznacza przepływ w łuku e wówczas w sieci z mnożnikami spełniane są następujące warunki:

$$0 \leq \varepsilon_{(p,m)} \leq E_{(p,m)} \tag{2.13}$$

$$\forall_{p \neq t} \forall_{p \neq s} \quad \eta_p \sum_{m \in \Gamma^{-1}_p} \varepsilon_{(m,p)} - \sum_{m \in \Gamma_p} \varepsilon_{(p,m)} = 0 \tag{2.14}$$

UWAGA: Oprócz takiego określenia sieci można spotkać inne, ogólniejsze, w którym mnożniki przyporządkowane są łukom [39,66,67,68]. Tu tak ogólne sformułowanie nie jest potrzebne.

W sieci z mnożnikami rozpatrywane będzie zadanie szukania przepływu o minimalnym koszcie, które można sformułować w sposób następujący:

znaleźć takie liczby całkowite $\overset{*}{\varepsilon}_{(p,m)}$, które minimalizują:

$$\sum_{(p,m) \in A} \varepsilon_{(p,m)} \cdot \lambda_{(p,m)}$$

i spełniają ograniczenia /2.13./ oraz /2,14/

oraz dodatkowo

$$\sum_{p \in \Gamma_s} \varepsilon_{(s,p)} = v_s$$

albo

$$\sum_{p \in \Gamma^{-1}_t} \varepsilon_{(p,t)} = v_t$$

UWAGA: gdy wszystkie liczby $\lambda_e \leq 0$ wówczas można zrezygnować z powyższych warunków.

Algorytm konstruowania sieci równoważnej zadaniu /2.12/

Dla potrzeb dalszych rozważań każdy z programów podzielony zostanie na M_n^i zadań o jednostkowym czasie trwania. Wprowadzimy następujące oznaczenie: j^i oznacza j-te zadanie $1 \leq j \leq M_n^i$ i-tego programu. Do wykonania, zadanie j^i wymaga $b_n^i(j)$ jednostek zasobów SC. Zadania muszą być wykonywane w ściśle określonej kolejności tzn. dla każdego i najpierw zadanie 1^i później 2^i, itd, na końcu zadanie $j^i, j=M_n^i$.

Aby jednoznacznie określić sieć zdefiniujemy zbiór $\mathcal{N}$ i zbiór $\mathcal{A}$ oraz podamy zasady określania mnożników.

Przepustowości wszystkich łuków $E_e = 1, \forall e \in \mathcal{A}$

Zbiór $\mathcal{N}$ składa się z dziewięciu rozłącznych podzbiorów wierzchołków. Wierzchołki każdego podzbioru mają określone znaczenie związane z zadaniem rozdziału zasobów.

1° Podzbiór pierwszy N^1 składa się z jednego wierzchołka będącego źródłem sieci s.

2° Na podzbiór drugi N^2 składa się T wierzchołków, każdy z nich odpowiada jednemu z odcinków czasu $[k,k+1)$, $k=0,1,\ldots,T-1$
Dla każdego $p \in N^2$ istnieje łuk (s,p), łączący go ze źródłem.
$\forall\, p \in N^2$, $\eta_{sp} = 1$

3° W podzbiorze trzecim N^3 znajdują się wierzchołki odpowiadające możliwości równoczesnego wykonywania zadań różnych programów.

Jeżeli $J_p \neq \phi$ jest zbiorem zadań różnych programów, które mogą być wykonywane równocześnie tzn.

$$\sum_{j^i \in J_p} b^i(j) \leq c$$

to w N^3 znajduje się para węzłów p,p' zwanych dalej węzłami stowarzyszonymi i oznaczana $[p,p']$. Węzły te połączone są łukiem $(p,p') \in \mathcal{A}$.

Łatwo zauważyć, że moc zbioru N^3 spełnia następującą nierówność:

$$2 \cdot \sum_{i=1}^{I} M_n^i \leq N^3 \leq 2 \cdot \prod_{i=1}^{I} (M_n^i + 1)$$

Zdefiniujemy następujące dwa zbiory

J_p^1 zbiór zadań z J_p mających poprzedniki tzn.:

$$J_p^1 = \{ j^i \in J_p : j > 1 \}$$

J_p^2 zbiór zadań z J_p mających następniki, tzn.:

$$J_p^2 = \{ j^i \in J_p : j < M_m^i \}$$

Dla każdej pary stowarzyszonej $[p,p']$, $p \in N^3$, $p' \in N^3$ mnożniki określone są w następujący sposób:

$$\eta_p = \frac{1}{\bar{J}_p^1 + 1} \quad , \quad \eta_{p'} = \bar{J}_p + \bar{J}_p^2$$

Zbiory N^4, N^5, N^6 określone są w sposób następujący:

dla każdej pary stowarzyszonej $[p,p']$, $p,p' \in N^3$ istnieje dokładnie jeden $p'' \in N^4$ o mnożniku $\eta_{p''} = 1$ i taki, że

$$\forall r \in N^2 \; . \; (r, p'') \in A$$

5° dla każdego $j^i \in J_p^1$ istnieje dokładnie jeden

$p^{ji} \in N^5$ o mnożniku $\eta_{p^{ji}} = 1$ i taki, że

$$(p^{ji}, p) \in A$$

6° dla każdego $j^i \in J_p^2$ istnieje dokładnie jeden

$\bar{p}^{ji} \in N^6$ o mnożniku $\eta_{\bar{p}^{ji}} = 1$ i taki, że

$$(\bar{p}, p^{ji}) \in A$$

Pomiędzy węzłami ze zbiorów N^5 i N^6 są łuki określone w sposób następujący:

dla każdej pary stowarzyszonej $[p,p']$, $[r,r']$; $p,p',r,r' \in N^3$ i takiej, że dla każdego i jeżeli $j^i \in J_p$ a $k^i \in J_r$ oraz $k=j+1$, istnieje łuk $(\bar{p}^{ji}, r^{ki}) \in A$, łuki te odpowiedzialne są za kolejność wykonywania zadań.

7° do N^7 należą wierzchołki odpowiadające poszczególnym podzadaniom, jest ich $\sum_{i=1}^{M} M_n^i$, każdy z nich ma mnożnik 1.

$$\forall j^i \; \exists! \; \bar{\bar{p}}^{ji} \in N^7 \quad j^i \in J_p \Rightarrow (p', \bar{\bar{p}}^{ji}) \in A$$

8^o W N^8 znajduje się I węzłów każdy odpowadający jednemu z programów, o mnożniku $1/M_n^i$

każdy z wierzchołków z N^8 połączony jest łukami z węzłami z N^7 odpowiadającymi zadaniom tego programu.

$$\forall p^i \in N^8 \quad (p^{j_i}, p^i) \in A \qquad , j=1,2,\ldots,M_n^i$$

9 Źródło t stanowi jedyny element N^9

$$\forall p^i \in N^8 \quad (p^i, t) \in A$$

PRZYKŁAD.

Rozpatrzmy następujący, prosty przykład, na którym zilustrujemy zasadę budowy sieci.

Niech $I=3$, $T=3$, $r=2$, $c=\begin{bmatrix}1\\1\end{bmatrix}$, $M_n^1=2$, $M_n^2=2$, $M_n^3=1$

$b_n^1(1)=\begin{bmatrix}1\\0\end{bmatrix}$ $\quad b_n^2(1)=\begin{bmatrix}1\\1\end{bmatrix}$ $\quad b_n^3(1)=\begin{bmatrix}1\\0\end{bmatrix}$

$b_n^1(2)=\begin{bmatrix}0\\1\end{bmatrix}$ $\quad b_n^2(2)=\begin{bmatrix}0\\1\end{bmatrix}$

Równocześnie mogą być wykonywane następujące zadania 1^1 i 2^2, 1^2 i 1^3, 2^2 i 1^3.

Sieć odpowadająca temu zadaniu przedstawione jest na rysunku 2.4.

$N^1 = \{0\}$
$N^2 = \{1,2,3\}$
$N^3 = \{11,11',12,12',13,13',14,14',15,15',16,16',17,17'\}$
$N^4 = \{4,5,6,7,8,9,10\}$
$N^5 = \{18,19,20,21\}$
$N^6 = \{22,23,24\}$
$N^7 = \{25,26,27,28,29\}$
$N^8 = \{30,31,32\}$
$N^9 = \{33\}$

	mnożniki
$[11,11']$ odpowiada wykonaniu zadania 1^1	1,2
$[12,12']$ odpowiada wykonaniu zadania 1^2	1/2, 1

		mnożniki
[13,13']	odpowiada wykonaniu zadania $1^1, 2^2$	1/2, 3
[14,14']	odpowiada wykonaniu zadania 2	2, 1
[15,15']	odpowiada wykonaniu zadania 2	1/2, 1
[16,16']	odpowiada wykonaniu zadania $2^2, 1^3$	1/2, 2
[17,17']	odpowiada wykonaniu zadania 1^3	1, 1

18 - węzeł odpowiadający poprzednikowi zadania 1^2 w węźle 12

19,20,21 węzły odpowiadające - " - - " - 2^2 w węzłach 13,15,16

22,23 - węzły odpowiadające następnikowi zadania 1^1 w węzłach 11,13

24 - węzeł odpowiadający - " - -" - 2 w węźle 14

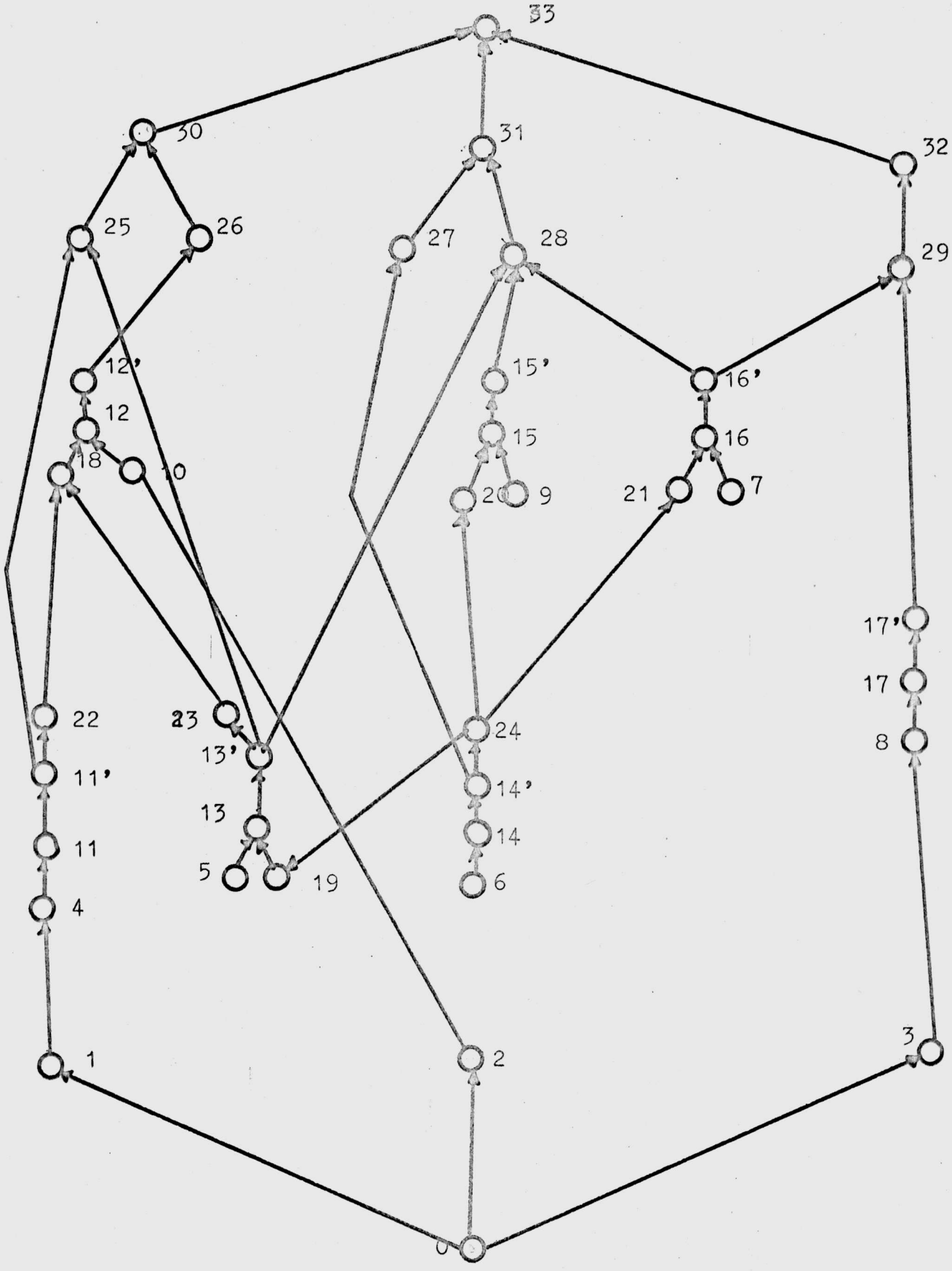
25,26 - węzły odpowiadające zadaniom programu 1

27,28 - węzły odpowiadające zadaniom programu 2

29 - węzeł odpowiadający zadaniu programu 3

		mnożnik
30	- węzeł odpowiadający programowi 1	1/2
31	- węzeł odpowiadający programowi 2	1/2
32	- węzeł odpowiadający programowi 3	1

UWAGA: na rysunku, aby go nie zaciemniać, nie umieszczono wszystkich łuków pomiędzy węzłami zbioru N^2 a N^4. □



Rys. 2.4. Sieć z przykładu.

Interpretacja przepływu w sieci z mnożnikami

Pokażemy teraz w jaki sposób należy interpretować dowolny przepływ całkowitoliczbowy w sieci z mnożnikami, skonstruowanej według podanego algorytmu.

Rozpatrzmy dowolny, niezerowy przepływ spełniający ograniczenia /2.14/.

Przepływ jednostkowego strumienia przez węzeł należący do N^2 oznacza przydział jednostki czasu pewnej grupie zadań. Określić tę grupę można sprawdzając do którego węzła z N^4 przepływ ten został skierowany.

Jednostkowy strumień przepływający przez węzeł należący do N^8 wskazuje, że program odpowiadający temu węzłowi został uszeregowany.
Sumaryczny strumień wpływający do ujścia określa liczbę uszeregowanych programów, a sumaryczny strumień wypływający ze źródła czas potrzebny na uszeregowanie tych programów.

Jednostkowy strumień przepływający przez węzeł z N^7 wskazuje, że zadanie odpowiadające temu węzłowi zostało uszeregowane.

Nasycenie łuku pomiędzy węzłami pary stowarzyszonej $[p, p']$ oznacza, że zadania należące do J_p wykonywane będą równocześnie.

Ponieważ wierzchołki z N^2 odpowiadające taktom rozdziału zasobów, nie są i nie mogą być przyporządkowane w sposób jednoznaczny konkretnym taktom, z przepływu nie można wyznaczyć w sposób bezpośredni kolejności w jakiej mają być wykonywane poszczególne zbiory zadań .

Dalej podany zostanie sposób takiego przekształcenia sieci aby można uzyskać rozwiązanie i tego problemu.

W tym celu należy dokonać redukcji grafu, polega ona na:

- odrzuceniu wszystkich łuków o przepływie zerowym,
- odrzuceniu wszystkich izolowanych wierzchołków,
- odrzuceniu źródła ujścia oraz wszystkich węzłów z N^2, N^4

N^7, N^8 oraz likwidacji wszystkich łuków "pozbawionych" węzła początkowego lub końcowego,

- likwidacji łuków łączących węzły pary sprzężonej oraz zastąpieniu pary węzłów jednym węzłem,
- zastąpieniu dróg pomiędzy węzłami zmodyfikowanego zbioru N^3 prowadzących przez węzły z N^5 i N^6 przez łuki bezpośrednie.

W tak przekształconym grafie pozostały jedynie węzły odpowiadające zbiorom zadań, które mogą być wykonywane równocześnie wraz z łukami zapewniającymi zachowanie odpowiedniej kolejności.

Tak skonstruowany graf należy uporządkować stosując zaczerpnięty z [35] algorytm.

Uporządkowaniem grafu $G = \{N, A\}$ będziemy nazywali rozbicie zbioru jego wierzchołków N na podzbiory niepuste $N_1, N_2, \ldots, N_m$, takie że $\bigcup_{i=1}^{m} N_i = N$, $N_i \cap N_j = \phi$ $i \neq j$ oraz

- jeżeli $p \in N_i$, $r \in N_j$ i $(p, r) \in A$ to $j > i$
- $\Gamma_p^{-1} = \phi$ i $\Gamma_p \neq \phi \Rightarrow p \in N_1$
- $\Gamma_p = 0$ to $p \in N_m$

Następujący algorytm dokonuje uporządkowania grafu G:

1° Do zbioru N_1 zaliczamy wszystkie wierzchołki p grafu, dla których $\Gamma_p^{-1} = \phi$

2° Do zbioru N_k $(k = 2, \ldots, m-1)$ zaliczamy wszystkie te wierzchołki p grafu, dla których $\Gamma_p^{-1} \subset N_{k-1}$ i, $\Gamma_p \neq \phi$

3° Do zbioru N_m - wszystkie te wierzchołki p, dla których $\Gamma_p = \phi$

Po dokonaniu uporządkowania zredukowanego grafu można ustalić kolejność wykonywania zadań w następujący sposób:

- jeżeli $p \in N_k$ i $r \in N_{k'}$ to
- jeżeli $k' > k$ zadania odpowiadające p powinny być wykonywane wcześniej niż zadania odpowiadające r,

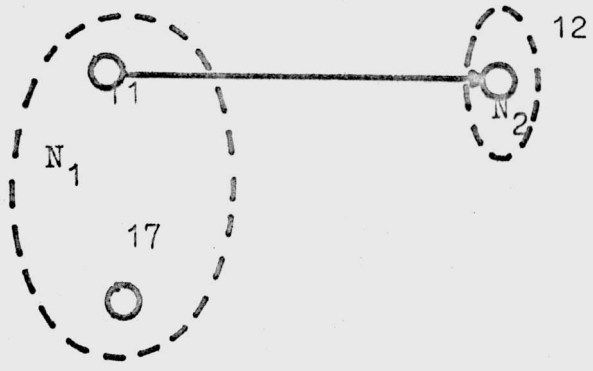
- jeżeli $p \in N_k$ i $\tau \in N_k$ zadania odpowiadające p i zadania odpowiadające τ mogą być wykonywane w dowolnej kolejności.

PRZYKŁAD, c,d.

W grafie z rysunku 2.4 rozpatrzmy następujący przepływ:
Na łukach (0,3), (3,4), (4,11), (11,11'), (11',25), (11',22), (22,18), (0,2), (2,10), (10,12), (18,12), (12,12'), (12',26), (25,30), (26,30), (30,33), (0,1), (1,8), (8,17), (17,17'), (17',29), (29,32), (32,33) przepływ równy 1, na pozostałych łukach 0.

Przepływ ten odpowiada wykonaniu zadań 1^1, 2^1, oraz 1^3. Uszeregowanie to wymaga trzech jednostek czasu i odpowiada wykonaniu programów o numerach 1 i 3.

Po redukcji graf przyjmie postać przedstawioną na rysunku 2.5. Można tam wyróżnić dwie klasy $N_1 = \{11, 17\}$ i $N_2 = \{12\}$, zadania 1^2 i 1^3 mogą być wykonane w dowolnej kolejności, w taktach 0 i 1, a zadanie 2^1 w takcie 2. □



Rys. 2.5. Graf z rysunku 2.4 po redukcji.

Dowód poprawności konstrukcji sieci

Pokażemy teraz, że dowolny przepływ całkowitoliczbowy zinterpretowany według wskazówek podanych wcześniej stanowi uszeregowanie dopuszczalne.

Łatwo zauważyć, że w zaproponowanej sieci zawsze istnieje niezerowy przepływ całkowitoliczbowy - jest nim na przykład przepływ odpowiadający uszeregowaniu jedynie programu pierwszego. W tym sensie konstrukcja jest poprawna.

Przez uszeregowanie dopuszczalne rozumiemy takie uszeregowanie, w którym nie są naruszone ograniczenia na zasoby ani na kolejność wykonywania zadań, żadne zadanie nie jest wykonane dwukrotnie, jeżeli z interpretacji przepływu wynika, że został uszeregowany program, to zostały uszeregowane wszystkie jego zadania.

Rozpatrzmy kolejno wszystkie powyższe warunki.

Z konstrukcji par węzłów z N^3 wynika, że ograniczenia na zasoby będą spełnione zawsze. Z doboru mnożników węzłów stowarzyszonych wynika następujący fakt: aby nasycić łuk (p,p') pary $[p,p]$ do węzła p musi wpłynąć strumień o wielkości $1+\bar{J}_p^1$ przez tyleż łuków, każdy o przepustowości 1.
Jednostkowy przepływ wpłynie od dokładnie jednego węzła ze zbioru N^2 /ze względu na istnienie węzła z grupy N^4/, pozostały przepływ wpłynąć musi od poprzedników zadań w J_p.
Z węzła p' wypływa zawsze strumień o wielkości $\bar{J}_p^2+\bar{J}_p$ przez tyleż łuków. Część przpływu kierowana jest do węzłów z N^7 informując o wykonaniu odpowiednich zadań i nasycając odpowiednie łuki pomiędzy N^7 a N^8 /w związku z tym żadne zadanie nie zostanie uszeregowane dwukrotnie/. Pozostała część przepływu skierowana zostaje do następników zadań.
Program i zostanie uznany za uszeregowany tylko wtedy, gdy przepłynie jednostkowy przepływ przez odpowiadający wierzchołek ze zbioru N^8, a nastąpi to tylko wtedy, gdy do tego wierzchołka wpłynie przepływ równy M_n^i.

Zauważmy również, że w każdym konkretnym przypadku można usunąć z sieci pewne węzły, nie tracąc przy tym pokazanych wcześniej właściwości przepływu. Dotyczy to szczególnie węzłów z N^5 i N^6. Podamy tu ogólne wskazówki tego postępowania: ze zbiorów N^5 i N^6 odrzucić można te węzły, do których wchodzi i wychodzi dokładnie jeden łuk, skorygować wówczas należy inne łuki.

Na przykład w sieci z rys. 2.4. można odrzucić węzeł 20 i łuki (24,20) , (20,15) zastępując je łukiem (24,15) .

Zadanie przepływu o minimalnym koszcie

Pokażemy teraz w jaki sposób można przenieść zadanie (2.12) na zadanie przepływu o minimalnym koszcie w sieci z mnożnikami. Skorzystamy w tym celu z równoważnej postaci kryterium (2.8).

Jednostkowy przepływ na łuku pomiędzy węzłem p^i z N^8 a t oznacza wykonanie pewnego programu, przypisujemy temu łukowi koszt $\lambda_{(p^i,t)} = \delta^i(z_n^i)$. Teraz sumaryczny koszt strumienia wpływającego do t równy jest wartości wskaźnika jakości odpowiadającego wykonaniu wybrany programów.

Oznaczmy przez $A^1 = \{(p,t) \in \mathcal{A} : p \in N^8\}$

Przyjmijmy $\lambda_e = 0 \quad e \notin A^1$ oraz

$$\lambda_{(p^i,t)} = \delta^i(z_n^i), \quad (p^i,t) \in A^1$$

Jak pokazano, wyjściowy problem/2.12/ równoważny jest następującemu zadaniu szukania całkowitoliczbowego przepływu o minimalnym koszcie w sieci z mnożnikami:

Znaleźć przepływy $\varepsilon^*_{(p,m)}, (p,m) \in \mathcal{A}$ minimalizujące

$$\sum_{p \in N^8} \varepsilon_{(p,t)} \cdot \lambda_{(p,t)}$$

przy ograniczeniach:

$$\forall_{\substack{p \neq s \\ p \neq t}} \quad \eta_p \sum_{m \in \Gamma_p^{-1}} \varepsilon_{(m,p)} - \sum_{m \in \Gamma_p} \varepsilon_{(p,m)} = 0$$

$\varepsilon_{(p,m)}$ binarne

W ten sposób pewne nieliniowe zadanie programowania binarnego zostało przekształcone do liniowego programowania w zmiennych binarnych.
Uczyniono to kosztem zwiększenia wymiarowości problemu i wzrostu liczby ograniczeń. Postępowanie takie wydaje się uzasadnione ponieważ metody liniowego programowania całkowitoliczbowego są bardzo dobrze opracowane. Jest też wiele algorytmów rozwiązywania takich zadań.

Niestety, dla sieci z mnożnikami nie ma jak dotąd algorytmów poszukiwania przepływów całkowitoliczbowych o minimalnym koszcie uwzględniających specyfikę sieci. Wszystkie opracowane dotąd algorytmy [18,39,66,67,68] poszukują przepływu bez uwzględnienia warunku całkowitoliczobowości.

Można również bardzo łatwo pokazać, że skonstruowana sieć, nie może być zredukowana do sieci z jednostkowymi mnożnikami [66] .

Tak więc do szukania optymalnego przepływu w sieci należy używać klasycznych metod programowania całkowitoliczbowego [29,46] .

Jeżeli koszty na łukach z A' przyjmiemy równe -1, wówczas szukanie przepływu o minimalnym koszcie równoważne jest zadaniu szukania maksymalnego przepływu.
Problem ten może być wykorzystany do znajdowania uszeregowania przed liniami krytycznymi [8,9] .

Na zakończenie dodać należy, że wyżej przedstawiony algorytm budowy sieci został zainspirowany algorytmem przedstawionym w [8,9] .
Stanowi on istotne uogólnienie zaprezentowanych w [8] idei na przypadek zadań zależnych wykorzystujących różne zasoby.

Omówiony algorytm może być łatwo rozszerzony na przypadek ogólniejszych ograniczeń kolejnościowych niż potrzebne w tej pracy. Wystarczy zmodyfikować jedynie zasoby tworzenia zbiorów N^3, N^5 i N^6 i ustalania mnożników wierzchołków z N^3.

PRZYKŁAD c.d.

Niech w grafie z rys. 2.4 $\lambda_{(31,32)} = -5$, $\lambda_{(29,32)} = -3$, $\lambda_{(30,32)} = -2$

łatwo sprawdzić, że zarówno przepływ o nasyconych łukach
(0,1) , (1,6) , (6,14) , (14,14') , (14',24) , (24,20) , (20,15) , (15,15'), (15',28) , (28,31) , (31,33) , (0,2) , (2,9) , (9,15) , (0,3) , (3,8) , (8,17) , (17,17') , (17',29) , (29,32) (32,33) , (14,17) , (17,31)
jak i przepływ w którym łuki
(0,1) , (1,6) , (6,14) , (14,14') , (14',17) , (17,31) , (14,24) , (24,21) , (21,16) , (16,16') , (16',22) , (22,31) , (31,33) , (16',29) , (29,32) , (32,33) ,
są nasycone, osiąga minimalny koszt -8 i jest przepływem optymalnym.

Przy czym uszeregowanie odpowiadające drugiemu przepływowi wymaga 2 jednostek czasu, a odpowiadające pierwszemu przepływowi trzech. □

Na zakończenie podanych zostanie kilka uwag dotyczących realizacji uzyskanych procedur rozdziału zasobów. Na rysunku 2.6. zamieszczony jest bardzo uproszczony schemat blokowy tej części SO, która odpowiedzialna jest za uruchamianie programów. Jest on bardzo podobny do schematu z rysunku 2.1. różni się tylko znaczeniem poszczególnych bloków.

W przypadku gdy w każdym kroku sterowania programy wyliczające sterowania są jednakowe wówczas nie zmienia się struktura sieci /ograniczenia (2.14) są jednakowe/, zmieniają się jedynie koszty.
Działanie SO jest znacznie prostsze ponieważ ograniczenia mogą być wyznaczone wcześniej.

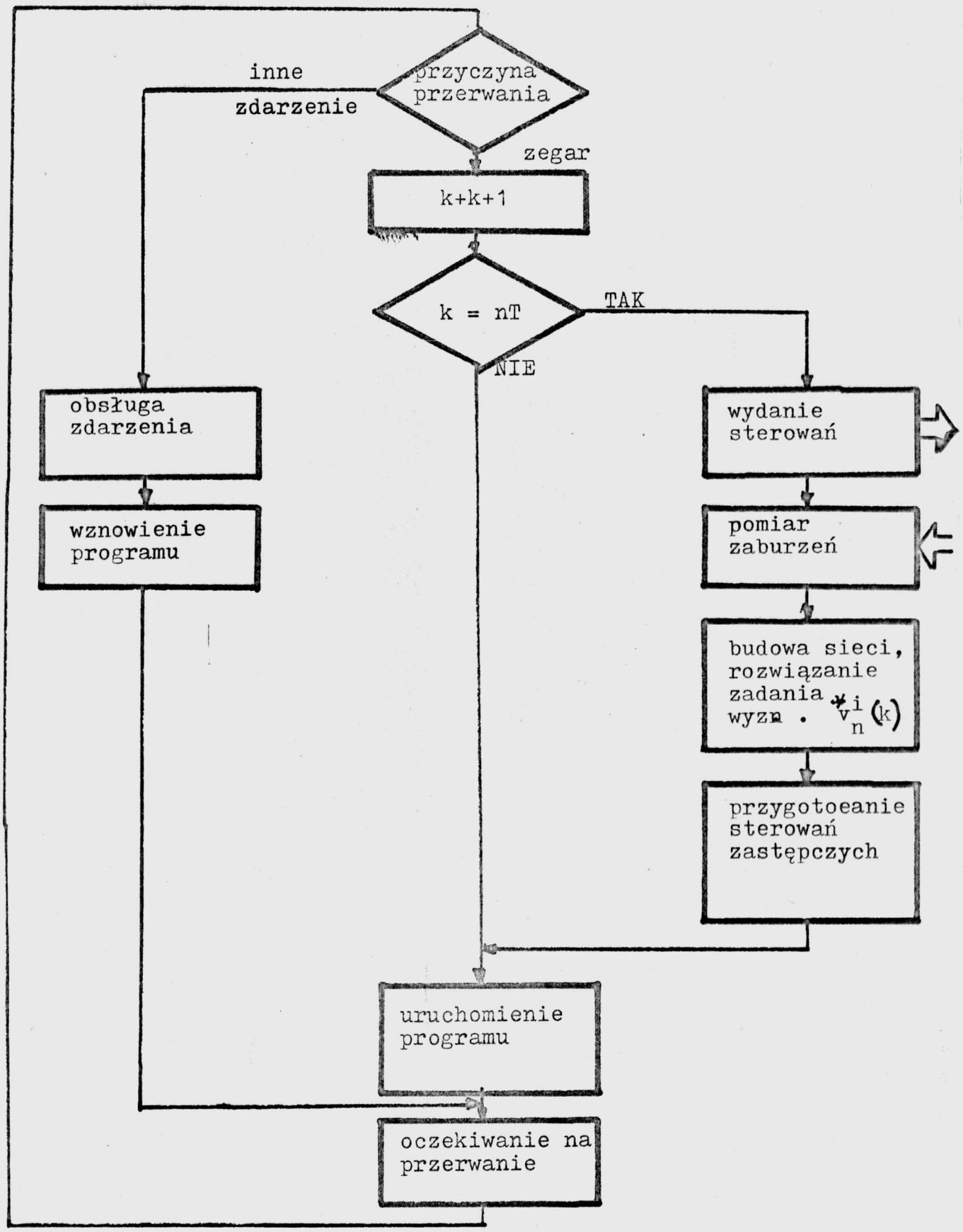
Dalsze uwagi dotyczące realizacji algorytmów optymalnego rozdziału zasobów podane zostaną w rozdziale 5.

Wszystkie uwagi dotyczące porównania reguł szeregowania powszechnie stosowanych z optymalnymi /w sensie kryterium (2.10)/ mogą być przeniesione z rozdziału 2.1.1.

Nieco inaczej wygląda natomiast sprawa możliwości rozbicia zadania na dwa problemy /por. 2.1.1. własność 3°: wyboru programów do wykonania i takiego ich szeregowania aby zostały wykonane przed linią krytyczną.

Ogólną własnością wszystkich zadań rozpatrywanych w tej pracy jest to, że kryterium zależy jedynie od faktu wykonania programów. Natomiast to ile programów może być wykonanych zależy w sposób istotny od kolejności wykonywania poszczególnych zadań. W związku z tym procedura rozdzielająca zasoby jest bardzo skomplikowana.

Stosunkowo niewiele prac poświęcono rozdziałowi procesora i dodatkowych zasobów [5,8,17,20] , tak, że trudno dokonać porównania proponowanych algorytmów z innymi.



Rys. 2.6. Uproszczony schemat blokowy procedury SO rozdziału zasobów.

3. ROZDZIAŁ ZASOBÓW W KOMPUTEROWYM SYSTEMIE STERUJĄCYM OBIEKTAMI DYNAMICZNYMI

Zajmiemy się teraz zadaniem rozdziału zasobów w KSS obiektami dynamicznymi. Podczas wszystkich rozważań obowiązywało będzie założenie, że długość kroku sterowania każdego obiektu jest jednakowa i wynosi T.

3.1. Zadanie rozdziału jednego zasobu.

3.1.1. Podstawowe własności zadania.

Rozpatrywane niżej zadanie ma bardzo wiele wspólnego z zadaniem z rozdziału 2.1.1.i dlatego,aby uniknąć powtórzeń będziemy odwoływać się do przeprowadzonych wcześniej rozważań. Podstawowa różnica pomiędzy zadaniem rozpatrzonym wcześniej a problemem w niniejszym rozdziale polega na tym, że w przypadku obiektów statycznych pogorszenie jakości sterowania w jednym kroku jest niezależne od decyzji w innych krokach sterowania. Natomiast w przypadku obiektów dynamicznych zależność taka występuje i stanowi istotne utrudnienie.

Problem optymalizacji rozdziału zasobów w KSS obiektami dynamicznymi zapisany być może w postaci:

znaleźć takie $v_n^{*i}(k)$;i=1,2,..,I;k=0,1,..,NT;n=0,1,..,N-1

które minimalizują Q'^{II}

$$Q'^{II} = \sum_{i=1}^{I} \alpha^i \sum_{n=0}^{N-1} \Phi_n^i \left(x_n^i, \gamma_n^i \left(\sum_{k=nT}^{(n+1)T-1} v_n^i(k), x_n^i \right) \right) \qquad (3.1)$$

gdzie

$$x_{n+1}^i = F_n^i \left(x_n^i, \gamma_n^i \left(\sum_{k=nT}^{(n+1)T-1} v_n^i(k), x_n^i \right) \right), \quad x_0^i \text{ - dane}$$

przy ograniczeniach:

$$\forall k \quad \sum_{i=1}^{I} v_n^i(k) \leq C, \quad n = \left[\frac{k}{T} \right] \qquad (3.2)$$

oraz wszystkie $v_n^i(k)$ binarne.

Przeprowadzając rozumowanie analogiczne jak w 2.1.1. i wprowadzając zmienne w_n^i zadanie można zapisać w postaci:

znaleźć $\overset{*i}{w}_n$ binarne, minimalizujące funkcję $\tilde{Q}^{II}$

$i=1,2,..,I$; $n=0,1,..,N-1$;

$$\tilde{Q}^{II} = \sum_{i=1}^{I} a^i \sum_{n=0}^{N-1} w_n^i \left[\Phi_n^i \left(x_n^i, \psi_n^i(x_n^i) \right) - \Phi_n^i \left(x_n^i, \tilde{u}_n^i \right) \right] + \\ + \tilde{\Phi}_n^i \left(x_n^i, \tilde{u}_n^i \right) \qquad (3.3)$$

przy ograniczeniach:

$$\forall_n \sum_{i=1}^{I} M_n^i \cdot w_n^i \leq c_T \qquad (3.4)$$

Liczby w_n^i, $i=1,2,..,I$; $n=0,1,..,N-1$ mają bardzo prostą interpretację. Określają one, który z programów w n-tym takcie ma być wykonany /por. 2.1.1./

Powyższe zadanie optymalizacji ma następujące własności:

- wartość kryterium nie zależy od kolejności wykonywania zadań, a jedynie od faktu ich wykonania
- rozwiązanie zadania może być przeprowadzone w dwu etapach:
  a/ - określenie, które z programów powinny być wykonane aby minimalizować kryterium nie przekraczając ograniczeń na zasoby SC,
  b/ - znaleźć takie uszeregowanie wybranych w punkcie a) programów aby zostały one wykonane w zadanym czasie,
- decyzje nie mogą być podejmowane niezależnie w każdym kroku sterowania,
- zadanie może być zaklasyfikowane jako zadanie sterowania dwupołożeniowego /wielkość sterująca, może przyjmować jedynie dwie wartości/ obiektem dynamicznym przy liniowych ograniczeniach na zmienne decyzyjne.

Sformułowane zadanie jest zadaniem typu Lagrange'a i jako takie spełnia warunek (3) z rozdziału 1.6 pracy [26], który zapewnia, że zasada optymalności Bellmana jest warunkiem koniecznym i dostatecznym optymalności.

W związku z tym można stosować medotę programowania dynamicznego do rozwiązania tego zadania.

Optymalne decyzje $\overset{*i}{w}_n$ mogą być uzyskane w dwu równoważnych postaciach: jako sterowania w układzie zamkniętym /funkcja

stanu obiektów/ lub jako sterowania w układzie otwartym /funkcja czasu/. Sposób realizacji algorytmu rozdziału zasobów zależeć będzie zawsze od konkretnej sytuacji. Pewne aspekty tego zagadnienia przedstawione zostaną w rozdziale 5.

ćW przypadku rozpatrywania zadania (1.6), (1,6') , możliwa jest tylko realizacja algorytmu optymalnego rozdziału zasobów w układzie zamkniętym.

Powstające i w tym przypadku zadanie szeregowania wybranych do wykonania programów rozwiązane być może za pomocą algorytmów zaproponowanych w 2.1.2. i nie będzie tu rozpatrywane.

Realizacja algorytmów optymalnego rodziału zasobów

Zasadniczy sposób realizacji w SO algorytmów optymalnego rozdziału zasobów jest taki sam jak pokazano na rysunku 2.1 Podstawowa różnica polega na tym, że w przypadku obiektów statycznych algorytm rozdziału zasobów był stacjonarny, w tym przypadku jest on niestacjonarny.

Inna różnica wynika z tego, że w przypadku obiektów statycznych, na każdym kroku należało rozwiązać zadanie optymalizacji. Natomiast w KSS obiektami dynamicznymi SO będzie realizował jedynie pewną statyczną zależności $w_n^{*i} = W_n^i(x_n^1, x_n^2, .., x_n^I)$, która powstała w wyniku wcześniejszego rozwiązania zadania optymalizacji. Problem został rozbity na dwa: wyboru optymalnej reguły rozdziału zasobów/może być wykonany wcześniej/ i realizacji optymalnej reguły /w czasie rzeczywistym/.

Na rysunku 3.1. podano uproszczony schemat blokowy procedury rozdziału Zasobów KSS obiektami dynamicznymi.

Dalsze uwagi na temat realizacji algorytmów rozdziału zasobów podano w rozdziale 5.

3.1.2. Algorytm rozdziału zasobów oparty na metodzie podziału i ograniczeń.

Zgodnie z rozważaniami poprzedniego podrozdziału rozpatrzymy jedynie zadanie (3.3) , (3.4) traktując problem szeregowania jako rozwiązany (por. 2.1.2.).

Mimo pozornej prostory otrzymanego zadania trudno znaleźć efektywną metodę jego rozwiązania. Może ono być, na przykład sprowadzone do statycznego zadania minimalizacji [10] . Takie podejście ma szereg wad i jest nieefektywne, z powodu ogromnej ilości zmiennych decyzyjnych, nieuwzględnienia specyfiki postaci ograniczeń (3.4.) /które są niezależne dla każdego kroku sterowania n/ oraz ograniczeń wynikających z postaci równań obiektu sterowania. Inną wadą takiego podejścia jest to, że otrzymane rozwiązanie zależy od stanu początkowego. Wad wymienionych wyżej pozbawiona jest metoda programowania dynamicznego. Jest to jedyna metoda dająca efektywne warunki szukania minimum w tym przypadku. Inne metody ([10]) mogą być stosowane gdy zbiór sterowań dopuszczalnych jest obszarem.

Podstawowymi zaletami metody programowania dynamicznego jest to, że:

- zasada optymalności Bellmana jest warunkiem koniecznym i dostatecznym optymalności,
- zadanie optymalizacji zostaje rozbite na szereg problemów o mniejszej wymiarowości,
- otrzymane reguły rozdziału zasobów realizowane są w układzie zamkniętym,
- algorytm programowania dynamicznego jest stosunkowo prosty w realizacji.

Do wad metody programowania dynamicznego oprócz dużego zapotrzebowania na pamięć zaliczyć można to, że w pewnych przypadkach, gdy trajektorie nie są "przemieszane" [28] jest ona bardzo nieefektywna.

Pokażemy teraz przebieg obliczeń z wykorzystaniem metody programowania dynamicznego.

Niech $B_n(x_n^1, x_n^2, \dots, x_n^I)$ oznacza funkcję Bellmana na $N - n + 1$ kroku obliczeń. Bn spełnia następujące równanie rekurencyjne:

$$B_n(x_n^1 \dots x_n^I) = \min_{w_n^i} \sum_{i=1}^{I} \alpha^i w_n^i \left[\Phi_n^i\left(x_n^i, \Psi_n^i(x_n^i)\right) - \Phi_n^i\left(x_n^i, u_n^i\right)\right] + \tag{3.5}$$

$$+ B_{n+1}\left(w_n^1 F_n^1\left(x_n^1, \Psi_n^1(x_n^1)\right) + (1 - w_n^1) F_n^1\left(x_n^1, \tilde{u}_n^1\right) \dots\right.$$

$$\ldots w_m^I F_m^I(x_m^I, \varphi_m^I(x_m^I)) + (1-w_m^I) F_m^I(x_m^I, \tilde{u}_m^I)) + \tilde{\Phi}_m^i(x_m^i, \tilde{u}_m^i)$$
$$m = N-1, N-2, \ldots, 1, 0,$$

(3.5 cd)

$$B_N(x_0^1, x_0^2, \ldots, x_0^I) = 0$$

$$\forall_m \sum_{i=1}^{I} M_m^i \cdot w_m^i \leq cT$$

Na każdym kroku obliczeń, "poruszając się od tyłu", dokonując minimalizacji otrzymujemy optymalną regułę rozdziału zasobów:

$$w_n^{*i} = W_n^i(x_n^1, x_n^2, \ldots, x_n^I) \qquad (3.6)$$

w układzie zamkniętym.

Regułę rozdziału zasobów w układzie otwartym można otrzymać przeprowadzając następujące obliczenia:

$$w_0^{*i} = W_0^i(x_0^1, x_0^2, \ldots, x_0^I)$$
$$x_{n+1}^i = w_n^i\left(F_n^i(x_n^i, \varphi_n^i(x_n^i)) + (1-w_n^i) F_n^i(x_n^i, \tilde{u}_n^i)\right.$$
$$w_{n+1}^{*i} = W_{n+1}^i(x_n^1, x_n^2, \ldots, x_n^I)$$
$$i=1,2,\ldots,I;\ n=0,1,\ldots,N-2$$

Otwarty pozostaje jedynie problem minimalizacji na każdym kroku procedury programowania dynamicznego. Zauważmy, że zadanie znalezienia w_{N-1}^i jest bardzo proste, dla każdego $x_{N-1}^1, x_{N-1}^2, \ldots, x_{N-1}^I$ dokonać należy mininalizacji funkcji:

$$\sum_{i=1}^{I} w_{N-1}^i \alpha^i \left[\Phi_{N-1}^i(x_{N-1}^i), \varphi_{N-1}^i(x_{N-1}^i)) - \tilde{\Phi}_{N-1}^i(x_{N-1}^i, \tilde{u}_{N-1}^i)\right] + \tilde{\Phi}_{N-1}^i(x_{N-1}^i, \tilde{u}_{N-1}^i)$$

przy ograniczeniach:

$$\sum_{i=1}^{I} w_{N-1}^i M_{N-1}^i \leq cT$$

Zadanie to jest dla każdego x_N^i zagadnieniem załadunku o zmiennych binarnych, i może być łatwo rozwiązane.

Na innych krokach programowania dynamicznego zadanie minimalizacji nie jest już tak proste - funkcja celu jest nieliniowa i nie może być w żaden sposób sprowadzona do postaci liniowej. Do rozwiązania tego zadania można stosować dowolną metodę programowania całkowitoliczbowego. Między innymi wykorzystać można metodę przeglądu leksykograficznego [29], która jest jednak bardzo nieefektywna.

Do rozwiązania tego problemu skorzystamy z metodyki tworzenia algorytmów typu podziału i ograniczeń.

Jak wiadomo [27] , skonstruowanie takiego algorytmu wymaga podania zasady podziału zbioru rozwiązań dopuszczalnych na podzbiory, sposobu wyznaczania oszacowań wartości funkcji celu na każdym z podzbiorów oraz skonstruowania reguły przeglądu podzbiorów.

Ponieważ metoda ta posiada bardzo obszerną literaturę np: [29,46] nie podamy jej dokładnego opisu ograniczając się jedynie do: określenia zasady podziału zbioru rozwiązań, szacowania wartości funkcji celu na podzbiorach i zaproponowania reguły przeglądu podzbiorów.

Ogólne zasady konstrukcji algorytmów opartych na metodzie podziału i oszacowań podano w [29] i stamtąd też zaczerpnięta jest terminologia używana w tej pracy.

Metoda podziału i ograniczeń należy do tych technik rozwiązywania zadań optymalizacji, które wykorzystują przegląd drzewa rozwiązań. Jest ono utworzone w następujący sposób. Korzeń drzewa h_o związany jest z wyjściowym zbiorem rozwiązań dopuszczalnych Wo.

Każda krawędź wychodząca z dowolnego wierzchołka odpowiada ograniczeniom nałożonym na jedną zmienną /ustala wartość jednej zmiennej/. Wierzchołek h_k odpowiada zbiorowi rozwiązań dopuszczalnych W_k w którym ustalono wartości zmiennych wynikających z krawędzi drogi P_k z h_o do h_k . Zbiór rowziązań dopuszczalnych W_{k+1} związany z następnikiem h_k, h_{k+1} jest oczywiście mniejszy od W_k ,tzn.: $W_k \supset W_{k+1}$

Wierzchołek h_k nazywać będziemy zamkniętym, gdy dalsze ustalanie wartości zmiennych z W_k nie ma sensu /nie mogą one spełniać ograniczeń/. W przeciwnym przypadku wierzchołek nazywany będzie aktywnym.

Dalsze rozważania dotyczą jednego kroku procedury programowania dynamicznego o numerze n , n= N-2,N-3,...,1,0.

Niech: $w_n = [w_n^1, w_n^2, \ldots, w_n^I]^T$

$x_n = [x_n^1, x_n^2, \ldots, x_n^I]^T$.

Oznaczmy funkcję, która podlega minimalizacji na n-tym kroku programowania dynamicznego $\tilde{B}_n(w_n)$:

$$\tilde{B}_n(w_n) = \sum_{i=1}^{I} w_n^i \alpha^i \left[\Phi_n^i(x_n^i, \Psi_n^i(x_n^i)) - \bar{\Phi}_n^i(x_n^i, \tilde{u}_n^i)\right] + \bar{\Phi}_n^i(x_n^i, \tilde{u}_n^i) + \tag{3.7}$$

$$+ \tilde{B}_{n+1}\left(w_n^1 \cdot F_n^1(x_n^1, \Psi_n^1(x_n^1)) + (1-w_n^1) F_n^1(x_n^1, \tilde{u}_n^1) \ldots\right.$$

$$\left. \ldots w_n^I \cdot F_n^I(x_n^I, \Psi_n^I(x_n^I)) + (1-w_n^I) F_n^I(x_n^I, \tilde{u}_n^I)\right)$$

Podamy teraz podstawowe elementy algorytmu budowanego w oparciu o metodę podziału i ograniczeń w przypadku zadania minimalizacji funkcji (3.7.)

Podział

Zbiór ograniczeń zadania optymalizacji zostanie zapisany jako

$$W_o = \left\{w_n : \sum_{i=1}^{I} w_n^i M_n^i \leq cT,\ w_n^i \text{ binarne}\right\}$$

Podział w wierzchołku h_k, $k=0,1,2,\ldots$ polega na wyborze pewnej zmiennej w_n^i, która nie była rozpatrywana na drodze P_k od h_o do h_k i na przyjęciu

$$W_k^* = \left\{W_k \cap \{w_n : w_n^i = 0\},\ W_k \cap \{w_n : w_n^i = 1\}\right\}$$

Drogę P_k tworzą krawędzie, które odpowiadają zmiennym o ustalonych wartościach zero lub jeden.
Podzbiór zmiennych o ustalonych wartościach /droga P_k z h_o do h_k nosi nazwę rozwiązania częściowego, przyjmijmy oznaczenie I_k dla zbioru indeksów zmiennych ustalonych oraz niech:

$$I_k^+ = \{i \in I_k : w^i = 1\},\quad I_k^- = \{i \in I_k : w^i = 0\}$$

$$\bar{I}_k = \{i : i \notin I_k\}$$

Proponowana metoda podziału jest typowa dla zadań optymalizacji ze zmiennymi binarnymi.

Oszacowania

Rozpatrzymy teraz problem oszacowania wartości $z_k = \min_{w_n \in W_k} \tilde{B}_n(w_n)$ z góry i z dołu

$$W_k = \{ w_n : w_n^i = 0, i \in \bar{I}_k^+, w_n^i = 1, i \in I_k^+, \sum_{i \in \bar{I}_k^+} w_n^i \cdot M_n^i \leq cT - \sum_{i \in I_k^+} M_n^i \} \quad (3.8)$$

Oszacowanie dolne oznaczać będziemy $\underline{z}_k$, a oszacowanie górne $\overline{z}_k$.

Podamy niżej dwie metody znajdowania oszacowań.

A. Jako oszacowanie dolne wartości z_k przyjmiemy liczbę $\underline{z}_k$ określoną w sposób następujący:

jeżeli: $cT - \sum_{i \in I_k^+} M_n^i = s < 0$

$$\underline{z}_k = +\infty \quad (3.9)$$

/zbiór W_k jest pusty/

jeżeli: $\forall i \; s < M_n^i$,

zbiór W_k zawiera tylko jeden element $\underline{w}$ określony w sposób następujący

$$\underline{w}^i = 1, \; i \in I_k^+$$
$$\underline{w}^i = 0, \; i \notin I_k^+$$

w tym przypadku (3.10)

$$\underline{z}_k = B(\underline{w})$$

W celu rozważania pozostałych przypadków wprowadzimy następujące oznaczenia:

niech $S_n^i(x_n^i)$ oznacza funkcję Bellmana zadania sterowania (1.4) tzn.:

$$S_n^i(x_n^i) = \min_{u_n^i \in U_n^i} \Phi_n^i(x_n^i, u_n^i) + S_{n+1}^i(F_n^i(x_n^i, u_n^i)) \quad n = N-1, N-2, \ldots, 1, 0$$

$$S_N^i(x_N^i) = 0$$

wówczas

$$\underline{z}_k = \sum_{i \in I_k^-} \alpha^i [\Phi_n^i(x_n^i, \tilde{u}_n^i) + S_{n+1}^i(F^i(x_n^i, \tilde{u}_n^i))] + \sum_{i \notin I_k^-} \alpha^i [\Phi_n^i(x_n^i, \psi_n^i(x_n^i)) + S_{n+1}^i(F^i(x_n^i, \psi_n^i(x_n^i)))] \quad (3.11)$$

$\underline{z}_k$ oznacza wartość wskaźnika jakości sterowania jaką można uzyskać sterując procesem technologicznym optymalnie /tzn. nie uwzględniając ograniczeń na zasoby/ w krokach n+1, n+2... N-1, a w kroku n-tym w sposób optymalny wszystkimi obiektami, których numery nie należą do I_k^-.

Natomiast obiektami o numerach $i \in \bar{I}_k$, sterujemy za pomocą sterowań zastępczych.

Jest oczywiste, że $\underline{z}_k \leq z_k$.

Inne oszacowania dolne uzyskamy w następujący sposób:

B. w przypadku gdy $s<0$ lub $s<M^i \ \forall i \in \bar{I}_k$

$\underline{z}'_k$ określamy jak $\underline{z}_k$ w przypadku A.

w przeciwnym razie rozważmy następujące zadanie optymalizacji

$$\underline{z}'_k = \min_{w} \sum_{i \in I} w^i \alpha^i \left[\Phi^i_n(x^i_n, \Psi^i_n(x^i_n)) - \Phi^i_n(x^i_n, \tilde{u}^i_n) + \right.$$
$$+ S^i_{n+1}(F^i_n(x^i_n, \Psi^i_n(x^i_n))) - S^i_{n+1}(F^i_n(x^i_n, \tilde{u}^i_n)) \Big] +$$
$$\left. + \alpha^i \left[\Phi^i_n(x^i_n, \tilde{u}^i_n) + S^i_{n+1}(F^i_n(x^i_n, \tilde{u}^i_n)) \right] \right] \qquad (3.12)$$

przy ograniczeniach

$$w \in W_k .$$

Ze względu na postać ograniczeń jest to liniowe zadanie załadunku o zmiennych binarnych.

$\underline{z}'_k$ oznacza minimalną wartość kryterium jaką można uzyskać sterując optymalnie wszystkimi obiektami w krokach n+1,n+2,... N-1 bez uwzględniania ograniczeń na zasoby /optymalnie/, a w kroku n uwzględniając je.

Dokonamy teraz krótkiego porównania obu metod wyliczania oszacowań. Oszacowania wyliczane według sposobu A są bardzo łatwe do uzyskania jeżeli założymy, że zadanie sterowania (1.4) w celu wyznaczenia algorytmów Ψ^i_n realizowanych przez KSS zostało rozwiązane przy użyciu metody programowania dynamicznego. W przypadku B do wyznaczania oszacowań dolnych, również wykorzystywane są funkcje Bellmana problemu (1.4), dodatkowo należy rozwiązać proste zadanie optymalizacji /zadanie załadunku/ w zmiennych binarnych.
Otrzymana w ten sposób wartość $\underline{z}'_k$ jest na ogół większa niż $\underline{z}_k$ co jest własnością pożądaną jeżeli chodzi o efektywność obliczeń.

Oszacowanie górne tzn. liczbę $\bar{z}_k \geq \min \tilde{B}_n(w)$, $w \in W_k$ uzyskamy w następujący sposób

$$\begin{aligned} &\text{niech} \quad \bar{w}_i = 1, \; i \in I_k^+ \\ &\bar{w}_i = 0, \; i \notin I_k^+ \\ &\bar{w} = [\bar{w}^1, \bar{w}^2, \ldots, {}^{wI}]^T, \; \bar{w} \in W_k \end{aligned} \tag{3.13}$$

Jako oszacowanie górne może być wzięta wartość funkcji $\tilde{B}_n$ w dowolnym punkcie należącym do W_k

Wybór zmiennej do podziału

Podamy teraz reguły wyboru zmiennej, której wartość zostanie ustalona na krawędziach wychodzących z h_k.

Dla każdego $i \in \bar{I}_k$ i takiego, że M_{m}^{i} wyliczamy Δ_k^i

$$\Delta_k^i = \tilde{B}_m(\bar{w}^{(i)}) - \tilde{B}_m(\bar{w})$$

gdzie $\bar{w}^{(i)} \in W_k$: $\bar{w}^{(i)j} = 0 \quad j \neq i \; j \in \bar{I}_k, \; \bar{w}^{(i)i} = 1$ (3.14)

$\bar{w} \in W_k$: $\bar{w}^j = 0 \quad j \in \bar{I}_k$

Δ_k^i oznacza zmianę kryterium gdy w n-tym kroku sterowania wykonamy i-ty program, przy ustalonych wartościach $i \in I_k$

Do podziału wybierzemy tę zmienną i° dla której $\Delta_k^{i^\circ}$ jest najmniejsze.

Załóżmy, że zawsze najpierw będziemy rozpatrywać $w^{i^\circ} = 1$ w takiej sytuacji droga P_k z h_o do h_k, jednoznacznie opisuje stan przeglądu drzewa rozwiązań [29].

Metody wyliczania oszacowań i wybierania zmiennej której wartość ma być ustalona mają zasadniczy wpływ na efektywność algorytmu. Dlatego też w każdym konkretnym przypadku warto poszukać lepszych oszacowań niż A i B. Zapewni to większą efektywność metody.

Algorytm metody podziału i ograniczeń

Krok 1 - W wierzchołku h_o: $I_o = 0$, $\underline{z}_o = -\infty$, $\bar{z}_o = +\infty$
$k = 1$. Przejdź do kroku 2.

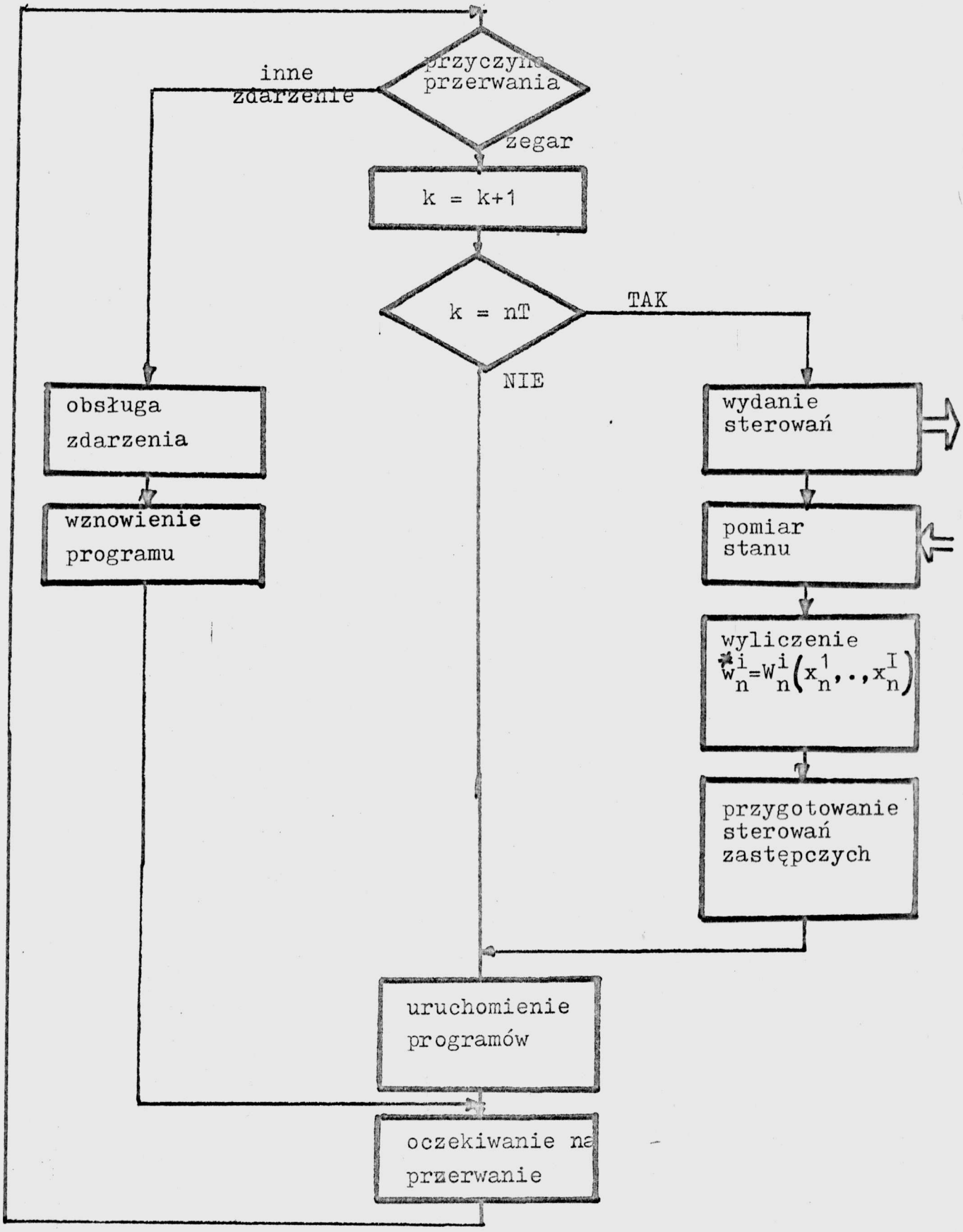
Krok 2 - obliczanie oszacowań. Wylicz oszacowanie w h_k na podstawie A /lub B/ i (3.13), Podstaw $\bar{z}_o = \min\{\bar{z}_o, \bar{z}_k\}$

Krok 3 - zamykanie wierzchołków. Jeżeli zachodzi warunek (3.9.) lub (3.10) lub $\bar{z}_k = \underline{z}_k$ lub $\underline{z}_k \geq \bar{z}_o$ zamknij h_k i przejdź do kroku 4. W przeciwnym razie przejdź do kroku 5.

Krok 4 - cofanie się. Jeżeli nie ma wierzchołków aktywnych przejdź do kroku 6. W przeciwnym przypadku wybierz gałąź prowadzącą do ostatnio otrzymanego wierzchołka aktywnego i przejdź do kroku 2.

Krok 5 - podział i gałęzienie. Dokonaj rozbicia wp (3.8) wybierając zmienną w^{i^o} zgodnie z (3.14). Wybierz gałąź $w^{i^o}=1$. Przejdź do kroku 3.

Krok 6 - końcowy. Jeżeli $\bar{z}_o=+\infty$ to zadanie nie ma rozwiązania dopuszczalnego, jeżeli $\bar{z}_o<\infty$ to rozwiązanie dopuszczalne, które daje wartość $\bar{z}_o$ jest rozwiązaniem optymalnym.



Rys. 3.1. Uproszczony schemat blokowy procedur rozdziału zasobów.

3.2. Zadanie rozdziału wielu zasobów

3.2.1. Własności rozwiązań otpymalnych

Zadanie /P2/ rozpatrywane w tym rozdziale może być zapisane w następującej postaci /por. 2.2.1./

znaleźć $v_m^i(k)$ minimalizujące Q^{II} , $i=1,2,..,I; n=0,1,..,N-1$

$$Q^{II} = \sum_{i=1}^{I} \alpha^i \sum_{n=0}^{N-1} \Phi_n^i\left(x_n^i, \gamma_n^i\left(\sum_{k=nT}^{(n+1)T-1} v_n^i(k), x_n^i\right)\right) \qquad (3.15)$$

przy ograniczeniach

$$x_{n+1}^i = F_n^i\left(x_n^i, \gamma_n^i\left(\sum_{k=nT}^{(n+1)T-1} v_n^i(k), x_n^i\right)\right), \quad x_0^i \text{ dane}, \; i=1,2,\ldots,I \qquad (3.16)$$

$$\forall_k \sum_{i=1}^{I} a_n^i(k) \cdot v_n^i(k) \le c \quad , n = \left[\frac{k}{T}\right]$$

$v_n^i(k)$ binarne,

γ_n^i określone jest wzorem /1.9/ a F_n^i i Φ_n^i wzorem /1.4/

Otrzymane zadanie sterowania nie jest łatwe do rozwiązania. Nie można tu rozdzielić problemu wyboru programów do wykonania od zadania ich szeregowania. Natomiast po rozwiązaniu zadania można wskazać programy, które zostaną wykonane w całości i te, które nie będą wykonane.

Realizacja algorytmu rozdziału zasobów będzie przebiegała w sposób następujący: po pomiarze stanu na początku kroku sterowania wyliczone zostaną decyzje SO na wszystkie T taktów pracy systemu. W kolejnych taktach SO będzie korzystał z wyliczonych wcześniej wartości $v_n^i(k)$,
Wykonywane jest to w sposób zbliżony do opisanego w 3.1.1. i nie będzie tu opisywane dokładniej.

Dalsze, ogólne uwagi na temat realizacji algorytmów optymalnego rozdziału zasobów podane będą w rozdziale 5.

Zauważmy, że zadania tego nie można rozbić na dwa problemy:

- wskazania programów, które powinny być wykonane aby efektywność KSS była maksymalna /jest to problem rozdziału zasobów na poziomie kroku sterowania obiektem/,

- takiego uszeregowania programów, aby mogły być one wykonane w zadanym czasie /problem szczegółowego rozdziału zasobów na poziomie taktów pracy SO/

mimo, iż wskaźnik jakości zależy jedynie od tego, które programy zostaną wykonane. Spowodowane to jest zmiennością zadań programów w czasie /podobna własność miało zadanie opisane w 2.2.1./

3.2.2. Metoda konstruowania algorytmu

Zadanie optymalizacji postawione w rozdziale 3.2.1. jest bardzo trudne do rozwiązania. W postawionej postaci może być ono atakowane pewną odmianą algorytmu opisanego w 3.1.2. /ze względu na binarność zmiennych/.

Innym, możliwym do zastosowania podejściem jest zastąpienie grupy ograniczeń /3.16/ odpowiadających jednemu n tzn. dla $nT \leq k < (n+1)T$, ograniczeniami liniowymi. Wykorzystać w tym celu należy algorytm opisany dokładnie w rozdziale 2.2.2. Funkcja celu może być bardzo łatwo sprowadzona do postaci liniowej poprzez wprowadzenie zmiennych w_n^i /por. 2.1.1. i 3.1.1./

Zadanie przyjmie w tym przypadku postać:

znaleźć liczby w_n^{*i} , i=1,2,..,I; n=0,1,...,N-1

minimalizujące

$$Q = \sum_{i=1}^{I} \alpha^i \sum_{n=0}^{N-1} w^i \left[\Phi_n^i(x_n^i, \varphi_n^i(x_n^i)) - \Phi_n^i(x_n^i, \tilde{u}_n^i) \right] + \Phi_n^i(x_n^i, \tilde{u}_n^i)$$

gdzie

$$x_{n+1}^i = w_n^i \left[F_n^i(x_n^i, \varphi_n^i(x_n^i)) - F_n^i(x_n^i, \tilde{u}_n^i) \right] + F_n^i(x_n^i, \tilde{u}_n^i)$$

dokonaliśmy tu utożsamienia przepływu na łuku (p_n^i, t_n), $p_n^i \in N_n^8$ /por. 2.2.2./ /p^i jest węzłem odpowiadającym wykonaniu programu i /zmienną w_n^i .

Wprowadziliśmy również dolny indeks "n" ponieważ teraz mamy do czynienia równocześnie z N sieciami.

Tak więc $\varepsilon_{(p_n^i, t_n)} = w_n^i$, a ograniczenia na przepływy mogą być zapisane

$$\forall n \; \forall p_n \in N_n \quad \eta_{p_n} \cdot \sum_{m \in \Gamma^{-1} p_n} \varepsilon_{(p_m, p_n)} - \sum_{m \in \Gamma p_n} \varepsilon_{(p_n, m)} = 0 \qquad (3.17)$$

$p_n \neq s_n$
$p_n \neq t_n$

Do rozwiązania powyższego zadania można podobnie jak w 3.1.2. zastosować metodę programowania dynamicznego. Na każdym kroku programowania dynamicznego zadanie do rozwiązania jest bardzo podobne jak w 3.1.2. z tą tylko różnicą, że tam było tylko jedno ograniczenie liniowe, tu jest ich wiele. Do rozwiązania tego zadania można stosować metodę podziału i ograniczeń opisana w 3.1.2. po dokonaniu jedynie niewielkich modyfikacji uwzględniających fakt, że jest wiele ograniczeń.

W pewnych sytuacjach, gdy liczba ograniczeń nie jest zbyt wielka można zastąpić ograniczenia /3.17/. jednym, liniowym i równoważnym im ograniczeniem /por. [29,46] /.

Podejście takie może być również stosowane do rozwiązywania zadania /2.12/ sprowadzając je do zagadnienia załadunku.

Redukcja zbioru ograniczeń liniowych do jednego ograniczenia możliwa jest tylko w przypadku stosunkowo niewielkiej liczby ograniczeń. Ponieważ gdy liczba redukowanych ograniczeń rośnie, rośnie również wielkość współczynników nowego ograniczenia.

Metoda ta z powodzeniem może być stosowania w omawianym przypadku, ponieważ współczynniki przy ograniczeniach /3.17/ są niewielkie, a zmienne decyzyjne są binarne co ma istotny wpływ na wielkość współczynników ograniczania wynikowego.

Algorytm redukcji wraz z podprogramem go realizującym opisany został dokładnie w Dodatku do niniejszej pracy [56] .

4. UOGÓLNIENIA

Poniżej podane zostaną uogólnienia zadań P1 i P2 rozpatrywanych w rozdziałach 2 i 3.

4.1. Różne okresy aktywizacji programów

Zbadamy teraz ogólniejsze zadanie rozdziału zasobów w KSS obiektami dynamicznymi. Uogólnienie polegać będzie na zrezygnowaniu z założenia o jednakowości długości kroków sterowania. Prowadzi to do skomplikowania zadania optymalizacji.

Określimy horyzont czasowy K zadania rozdziału zasobów. W przypadku zadań o jednakowej długości kroku sterowania, horyzont czasowy zadania rozdziału zasobów był taki sam jak horyzont czasowy wszystkich zadań sterowania. Teraz określimy go równym najdłuższemu horyzontowi zadania sterowania tzn.:

$$K = \max_i N^i T^i$$

Problem ogólniejszy od P2 sformułowany jest w sposób następujący:

znaleźć $\overset{*i}{v}_{n^i}(k), n^i=0,1,..,N^i-1;\ n^iT^i \le k < (n^i+1)T^i; i=1,2,..,I$

minimalizujące $Q^{"II}$

$$Q^{"II} = \sum_{i=1}^{I} \alpha^i \sum_{n^i=0}^{N^i-1} \Phi^i_{n^i}\left(x^i_{n^i}, \gamma^i_{n^i}\left(\varkappa^i_{n^i}((n^i+1)T^i), x^i_{n^i}\right)\right) \tag{4.1}$$

gdzie

$$x^i_{n^i+1} = F^i_{n^i}\left(x^i_{n^i}, \gamma^i_{n^i}\left(\varkappa^i_{n^i}((n^i+1)T^i), x^i_{n^i}\right)\right),\ x^i_0 \text{ dane}$$

$$\varkappa^i_{n^i}(k+1) = \varkappa^i_{n^i}(k) + v^i_{n^i}(k),\quad n^iT^i \le k < (n^i+1)T^i$$

$$\varkappa^i_{n^i}(k) = 0 \qquad k \le n^iT^i$$

przy ograniczeniach:

$$\forall k \sum_{i=1}^{I} a^i_{n^i}(k)\cdot v^i_{n^i}(k) \le c,\quad n^i = \left[\frac{k}{T^i}\right] \tag{4.2}$$

$$v^i_{n^i}(k) \text{ binarne}$$

gdzie:

n^i - numer kroku sterowania i-tym obiektem sterowania,

N^i - liczba kroków sterowania i-tym obiektem sterowania.

Tak postawione zadanie jest nieliniowym zadaniem sterowania optymalnego o binarnych zmiennych decyzyjnych.

Ze względu na "zachodzenie" na siebie taktów sterowania nie można wprowadzić zmiennych w^i, które tak ułatwiły rozwiązanie zadania opisanego w rozdziale 3.1.

Do rozwiązania zadania /4.1./, /4.2/ stosować będziemy metodę programowania dynamicznego.
Zalety i wady tej metody omówione zostały w rozdz. 3.1.2. i nie będą tu przypominane.
Metoda ta jest szczególnie użyteczna przy rozwiązywaniu problemów, w których ograniczenia zależą od stanu [28]. Ograniczenia /4.2/ mają własnie taką postać /por. rozdział 1.2./.

Problem optymalizacji powstający na każdym etapie programowania dynamicznego jest zadaniem o nieliniowej funkcji celu, liniowych ograniczeniach i binarnych zmiennych. Do jego rozwiązania, podobnie jak w 3.1.2. używany będzie algorytm oparty na metodzie podziału i ograniczeń.

Omówimy teraz dokładniej proponowaną metodę rozwiązywania zadania /4.1./, /4.2./.

Wprowadzmy następujące oznaczenia :

$$u(k)=\begin{bmatrix} u^1(k) \\ \vdots \\ u^I(k) \end{bmatrix}, \quad x(k)=\begin{bmatrix} x^1(k) \\ \vdots \\ x^I(k) \end{bmatrix}, \quad v(k)=\begin{bmatrix} v^1(k) \\ \vdots \\ v^I(k) \end{bmatrix}$$

gdzie

$$u^i(k)=u^i_{n^i}(k), \quad x^i(k)=x^i_{n^i}, \quad v^i(k)=v^i_{n^i}(k)$$

$$n^i=\left[\frac{k}{T^i}\right], \quad i=1,2,\ldots,I$$

Niech $\mathcal{J}_k=\{i: \exists_{n^i}\ k=n^i\cdot T^i\}$ oznacza zbiór numerów tych programów, dla których w chwili k kończy się jeden takt sterowania /o numerze n^i/, a zaczyna następny.

Funkcja Bellmana $B_k\left(x(k), u(k)\right)$ określona jest następującą zależnością rekurencyjną:

Jeżeli $\mathcal{J}_k \neq \phi$ wówczas

$$B_k(x(k), u(k)) = \min_{v(k) \in \mathcal{V}_k} B_{k+1}(x(k), u(k) + v(k)), \qquad (4.3')$$

a jeżeli $\mathcal{J}_k \neq \phi$

$$B_k(x(k), u(k)) = \min_{v(k) \in \mathcal{V}_k} B_{k+1}(\bar{x}(k), \bar{u}(k) + v(k)) + \sum_{i \in \mathcal{J}_k} \Phi^i_{m^i-1}\left(x^i(k), \gamma^i_{m^i-1}(u^i(k), x^i(k))\right) \qquad (4.3'')$$

gdzie

$$\mathcal{V}_k = \{ v(k) : \sum_{i=1}^{I} a^i_{m^i}(k) \cdot v^i(k) < e,\ v^i(k) \text{ binarne} \}$$

$$m^i = \left[\frac{k}{T^i}\right], \quad k = K, K-1, \ldots, 1, 0$$

$$B_{K+1}(x, u) = 0$$

$$\bar{x}(k) = \begin{bmatrix} \bar{x}^1(k) \\ \vdots \\ \bar{x}^I(k) \end{bmatrix} \qquad \begin{aligned} \bar{x}^i(k) &= x^i(k) \quad i \notin \mathcal{J}_k \\ \bar{x}^i(k) &= F^i_{m^i-1}(x^i(k), \gamma^i_{m^i}(u^i(k), x^i(k))), \quad i \in \mathcal{J}_k \end{aligned}$$

$$\bar{u}(k) = \begin{bmatrix} \bar{u}^1(k) \\ \vdots \\ \bar{u}^I(k) \end{bmatrix} \qquad \begin{aligned} \bar{u}^i(k) &= u^i(k) \quad i \notin \mathcal{J}_k \\ \bar{u}^i(k) &= 0 \quad i \in \mathcal{J}_k \end{aligned}$$

Oznaczmy funkcję podlegającą minimalizacji na każdym etapie programowania dynamicznego przez $\tilde{B}_k(v(k))$, tzn.:

$$\tilde{B}_k(v(k)) = B_{k+1}(x(k), u(k) + v(k)) \qquad \mathcal{J}_k = \phi$$

$$\tilde{B}_k(v(k)) = B_{k+1}(\bar{x}(k), \bar{u}(k) + v(k)) + \sum_{i \in \mathcal{J}_k} \Phi^i_{m^i-1}\left(x^i(k), \gamma^i_{m^i-1}(u^i(k), x^i(k))\right) \qquad \mathcal{J}_k \neq \phi$$

Omówimy teraz zaproponowany algorytm minimalizacji funkcji $\tilde{B}_k$. To nieliniowe zadanie programowania binarnego, mimo stosunkowo niewielkiej liczby zmiennych jest skomplikowane pod względem numerycznym. Do jego rozwiązania proponowany jest algorytm oparty na metodzie podziału i ograniczeń [29]. Omówione zostaną jedynie podstawowe elementy tego algorytmu wyliczenie oszacowań i wybór zmiennej do podziału. Dokładniej metoda podziału i ograniczeń omówione zostało w 3.2.2. Dalsze uwagi o algorytmach tego typu podane są w wielu monografiach, m.innymi [29, 44].

Sposób wyznaczania oszacowań

Wszystkie dalsze rozważania będą dotyczyły jednego, ustalonego etapu programowania dynamicznego. Wartości stanów obiektów $x^i(k)$ oraz stanu programów $\alpha(k)$ są ustalone. Podobnie jak w 3.1. do obliczania oszacowań używać będziemy funkcji Bellmana zadania sterowania /1.4/. Oznaczymy ją $S^i_{n^i}$ przyjmując definicję podaną w 3.1.2.

Zadanie minimalizacji w wierzchołku h_1 drzewa rozwiązań /por. 3.1.2./ określone jest w sposób następujący:

$$z_1 = \min_{v(k)\in V_1} \tilde{B}_k\left(v(k)\right)$$

zbiór rozwiązań dopuszczalnych V_1 określony jest następująco:

$$V_1 = \{v(k) : \sum_{i\in I_1} a^i(k)\cdot v^i(k) \le c - \sum_{i\in I_1^+} a^i(k),\ v^i(k)=0\ i\in I_1^-,\ v^i(k)=1\ i\in I_1^+\}$$

zbiory I_1, I_1, I_1^+, I_1^- określone zostały w 3.1.2.

Proponujemy następujący sposób wyznaczenia dolnego oszacowania $\underline{z}_1 \le z_1$.

Niech $$s = c - \sum_{i\in I_1^+} a^i(k) \qquad (4.4')$$

Jeżeli $s<0$ wówczas $V_1 = \phi$ i $z_1 = +\infty$

jeżeli $s\ge 0$ i $\forall_{i\in \bar{I}_1}\ a^i(k) > s$ wówczas $V_1 = \{\underline{v}\}$

$$\underline{v} = \begin{bmatrix} \underline{v}^1 \\ \vdots \\ \underline{v}^I \end{bmatrix}\quad \underline{v}^i = 1,\ i\in I_1^+;\ \underline{v}^i = 0,\ i\notin I_1^+,\quad \underline{z}_1 = \tilde{B}_k(\underline{v}) \qquad (4.4'')$$

w przeciwnym razie oszacowanie wyliczone będzie w następujący sposób:

niech $n^i = \left[\frac{k}{T^i}\right]$, $m^i = n^i - T^i - k$

$$\underline{z}_1 = \sum_{i=1}^{I} \alpha^i \left[\Phi^i_{m^i}\left(x^i_{m^i}, \gamma^i_{m^i}\left(x^i(k)+m^i, x^i_{m^i}\right)\right) + S^i_{m^i}\left(F^i\left(x^i_{m^i}, \gamma^i_{m^i}\left(x^i(k)+m^i, x^i_{m^i}\right)\right)\right)\right] \qquad (4.4''')$$

$\underline{z}_1$ wyliczone w ostatnim przypadku ma następującą interpretację: oznacza wartość wskaźnika jakości jaki może być osiągnięty w następujących warunkach:

- w krokach sterowania o numerach $n^i+1, n^i+2, .., N^i-1$; $i=1,2,..,I$ wydawane są sterowania optymalne,
- w kroku n^i wydawane jest sterowanie optymalne na i-ty obiekt jeżeli program wydający sterowanie zdąży zakończyć obliczenia startując ze stanu $x^i(k)$ w chwili k bez uwzględnienia ograniczeń na zasoby.

Jak łatwo zauważyć wartość ta jest zawsze mniejsza /lub równa/ niż najlepsza wartość wskaźnika jakości jaka może być osiągnięta, gdy w taktach k,k+1,..,K-1 uwzględnimy ograniczenia na zasoby.

Jako górne oszacowanie z przyjmiemy wartość $\tilde{B}_k$ w dowolnym punkcie z V_1 np.:

$$\bar{z}_l = \tilde{B}_k(\bar{\upsilon}) \qquad (4.5)$$

$$\bar{\upsilon} = \begin{bmatrix} \bar{\upsilon}^1 \\ \vdots \\ \bar{\upsilon}^I \end{bmatrix}, \quad \bar{\upsilon}^i = 1 \ \ i \in I_k^+, \ \bar{\upsilon}^i = 0 \ \ i \notin I_k^+$$

Wybór zmiennej do podziału

W celu wyboru w wierzchołku h_1 numeru zmiennej, której wartość będzie ustalona w procesie podziału, wyliczymy liczby Δ^i:

$$\forall i \in \bar{I}_k \wedge a^i(k) \leq s^i \quad \Delta^i = \tilde{B}_k(\upsilon^{(i)}) - \tilde{B}_k(\bar{\upsilon})$$

$$\upsilon^{(i)} = \begin{bmatrix} \upsilon^{(i)1} \\ \vdots \\ \upsilon^{(i)I} \end{bmatrix} \quad \upsilon^{(i)} \in V_l \quad \upsilon^{(i)j} = 0 \ \ j \neq i, \ j \in \bar{I}_k, \ \upsilon^{(i)i} = 1$$

$$\bar{\upsilon} = \begin{bmatrix} \bar{\upsilon}^1 \\ \vdots \\ \bar{\upsilon}^I \end{bmatrix} \quad \bar{\upsilon} \in V_l, \quad \bar{\upsilon}^i = 0 \ \ i \in \bar{I}_k.$$

Do podziału wybieramy zmienną o numerze i^0 takim, że

$$\Delta^{i^0} = \min_i \Delta^i \qquad (4.6.)$$

Ponieważ zarówno sposób wyliczania oszacowań jak i wyboru zmiennej do podziału mają zasadniczy wpływ na efektywność algorytmu, w każdym konkretnym przypadku warto poszukać lepszych sposobów ich wyznaczania.

Przebieg algorytmu

Poniżej podany zostanie algorytm oparty na metodzie podziału i ograniczeń dokonujący minimalizacji funkcji $\tilde{B}_k$ Zasadniczo nie różni się on od algorytmu zaproponowanego w 3.1.2. i nie będzie dokładnie omówiony.

Krok 1 - początkowy. W wierzchołku h_0: $I_0=\emptyset$, $\underline{z}_0 = -\infty$ $\bar{z}_0 = +\infty$, $l=1$. Przejdź do kroku 2.

Krok 2 - obliczanie oszacowań. Wylicz oszacowania w h_1 na podstawie /4.4./ i /4.5/. Podstaw $\bar{z}_0 = \min\{\bar{z}_0, \bar{z}_k\}$.

Krok 3 - zamykanie wierzchołków. Jeżeli zachodzi warunek (4.4) lub (4.4'') lub $\tilde{z}_1 = \underline{z}_1$ lub $\underline{z} \geq \tilde{z}$ zamknij h_{1i} przejdź do kroku 4. W przeciwnym razie przejdź do kroku 5.

Krok 4 - cofanie się. Jeżeli nie ma wierzchołków aktywnych przejdź do krok 5. W przeciwnym przypadku wybierz gałąź prowadzącą do ostatnio otrzymanego wierzchołka aktywnego i przejdź do kroku 2.

Krok 5 - podział i gałęzienie. Dokonaj rozbicia wg /3.8/ wybierając zmienną $v^{i^\circ}(k)$ zgodnie z /4.6/. Wybierz gałąź $v^{i^\circ}(k) = 1$. Przejdź do kroku 3.

Krok 6 - końcowy. Jeżeli $\tilde{z}_0 = +\infty$ to zadanie nie ma rozwiązania dopuszczalnego, jeżeli $\tilde{z}_0 < \infty$ to rozwiązanie dopuszczalne, które daje wartość jest rozwiązaniem optymalnym.

Realizacja algorytmu rozdziału zasobów w KSS

Poniżej omówione zostaną podstawowe własności rozwiązania optymalnego zadania /4.1./, 4.2/ oraz sposób realizacji algorytmu w KSS.

Rozważany problem jest znacznie bardziej skomplikowany niż zadanie rozważane w rozdziale 3.

Spowodowane jest to z jednej strony zmiennością żądań programów w czasie, a z drugiej strony faktem, że programy zgłaszają się do systemu w różnych chwilach.
Powoduje to, że decyzje SO wydawane być muszą na krótszy okres czasu na takt rozdziału zasobów a nie na krok sterowania.
Ma to również ogromny wpływ na realizację algorytmu w KSS - zwiększa się narzut wnoszony przez SO.

Algorytm rozdziału zasobów w zasadniczej swej części jest bardzo podobny do opisanego w rozdziale 3.1.1. na rys.3.1. i nie będzie dokładniej omawiany. Obowiązują również uwagi o dwuetapowym sposobie realizacji algorytmu: zadanie projektowania pewnego "regulatora" i zadanie realizacji algorytmu tego "regulatora".

Dalsze uwagi dotyczące realizacji podane zostały w rozdziale 5.

4.2. Różne linie krytyczne

Rozważania z rozdziału 2 prowadzone były przy założeniu, że wszystkie programy przychodzą do systemu /w jednym kroku sterowania/ w jednej chwili /co zwykle zapisuje się $r^i=0$/, a linia krytyczna jest jednakowa dla wszystkich programów. Teraz odejdziemy od drugiego założenia i przyjmiemy, że w każdym kroku sterowania określona jest dla każdego programu inna linia krytyczna d^i, $i=1,2,..,I$. Takie zadanie może być zapisane identycznie jak problem P1. Przeprowadzając rozumowanie analogiczne do przeprowadzonego w 2.1. powyższy problem można rozbić, w przypadku KSS obiektami statycznymi, na N niezależnych zadań.
Poniżej zajmiemy ją jednym takim zadaniem. Zrezygnujemy również, dla uproszczenia zapisu, z dolnego indeksu "n" występującego w zadaniu P1.

4.2.1. Rozdział jednego zasobu.

Zadanie optymalnego rozdziału jednego zasobu w KSS obiektami statycznymi w przypadku rozdziału jednej jednostki pojedyńczego zasobu zapisane być może w sposób następujący:

znaleźć $v^{*i}(k)$, $0 \le k < d^i$; $i=1,2,...,I$
minimalizujące Q

$$Q^I = \sum_{i=1}^{I} \alpha^i \cdot \varphi^i\left(f^i\left(\gamma^i\left(\sum_{k=0}^{d^i-1} v^i(k), z^i\right), z^i\right)\right) \tag{4.7}$$

przy ograniczeniach

$$\forall k \quad \sum_{i=1}^{I} v^i(k) \le 1, \quad v^i(k) \text{ binarne}, \; i=1,2,\ldots I \tag{4.8}$$

gdzie $\alpha^i, f^i, \varphi^i, \gamma^i, z^i$ mają znaczenie analogiczne do $\alpha^i_n, f^i_n, \varphi^i_n, \gamma^i_n, z^i_n$ używanych w 2.1.1.

d^i jest linią krytyczną /chwilą do której program musi zakończyć obliczenia/.

Dodatkowo przyjmiemy następujące oczywiste ograniczenie

$$v^i(k)=0, \quad k \geqslant d^i; \; i=1,2,\ldots,I,$$

oznaczające, że program nie może być wykonywany po przekroczeniu linii krytycznej.

Przeprowadzając rozumowanie analogiczne jak w rozdz. 2.1.2. wprowadzimy zmienne w^i

$$w^i = 1 \text{ jeżeli } \sum_{k=0}^{d^i-1} v^i(k) = M^i$$

$$w^i = 0 \text{ jeżeli } \sum_{k=0}^{d^i-1} v^i(k) < M^i$$

Wprowadzimy teraz ograniczenia równoważne /4.8/ na zmienne w^i.

Załóżmy najpierw, że

$$d^1 < d^2 < \ldots < d^I.$$

Jest oczywiste, że jeżeli

$$I_1 < I_2 < I$$

wówczas

$$\sum_{r=1}^{I_1} v^r(k) \leq \sum_{r=1}^{I_2} v^r(k) \leq \sum_{r=1}^{I} v^r(k)$$

Załóżmy, że spełnione jest ograniczenie /4.8/ wówczas, jak łatwo zauważyć, spełnione jest każde z następujących ograniczeń:

$$\sum_{k=0}^{d^j-1} \sum_{r=1}^{j} v^r(k) \leq d^j - 1 \qquad , j=1,2,\ldots,I$$

zmieniając kolejność sumowania otrzymamy

$$\sum_{r=1}^{j} \sum_{k=0}^{d^j-1} v^r(k) = \sum_{r=1}^{j} w^r M^r \leq d^j - 1 \qquad , j=1,2,\ldots,I \qquad (4.9)$$

Załóżmy teraz, że spełnione jest każde z ograniczeń (4.9) pokażemy, że spełnione jest ograniczenie /4.8/ przy dodatkowym założeniu

$$\sum_{k=0}^{d^j-1} v^r(k) = \begin{cases} 0 \\ M^r \end{cases} \text{lub}$$

Jest to oczywiste: j-te ograniczenie /4.9/ oznacza, że j pierwszych programów zostanie wykonanych do chwili d_j, j+1-sze ograniczenie oznacza, że jeżeli do j programów dołożymy jeszcze jeden, to obliczenia wszystkich programów dla których $w^i=1$ $i = 1,2,\ldots,j+1$ zostaną zakończone do chwili d^{j+1}.

Na rysunku 4.1. przedstawiono schemat blokowy algorytmu generacji $v^i(k)$ na podstawie w^i..

Funkcję celu /4.7/ można analogicznie jak w 2.1.1. zapisać w postaci zależnej od zmiennych w^i.

$$Q = \sum_{i=1}^{I} w^i \alpha^i \left[\varphi^i\left(f^i\left(\varphi^i(z^i), z^i\right)\right) - \varphi^i\left(f^i(\tilde{u}^i, z^i)\right)\right] + \alpha^i \varphi^i\left(f^i(\tilde{u}^i, z^i)\right)$$

i wprowadzając oznaczenia

$$\delta^i(z) = \alpha^i \left[\varphi^i\left(f^i\left(\varphi^i(z^i), z^i\right)\right) - \varphi^i\left(f^i(\tilde{u}^i, z^i)\right)\right]$$

i pomijając składniki $\alpha^i \varphi^i\left(f^i(\tilde{u}^i, z^i)\right)$

/nie mają one wpływu na położenie minimum/ zadanie /4.7/, /4.8/ może być zapisane w sposób następujący.

Znaleźć w^i, $i=1,2,..,I$ mininalizujące

$$Q = \sum_{i=1}^{I} w^i \delta^i(z^i) \qquad (4.10)$$

przy ograniczeniach

$$\sum_{i=1}^{j} w^i M^i \leq d^{j-1}, \qquad j=1,2,\ldots,I$$

$$w^i \text{ binarne}, \quad i=1,2,..,I \qquad (4.11)$$

Jest to dla każdego układu liczb z^i, $i=1,2,..,I$ zadanie liniowego programowania binarnego.

Zauważmy, że w przypadku gdy wartości d^i spełniają zależność

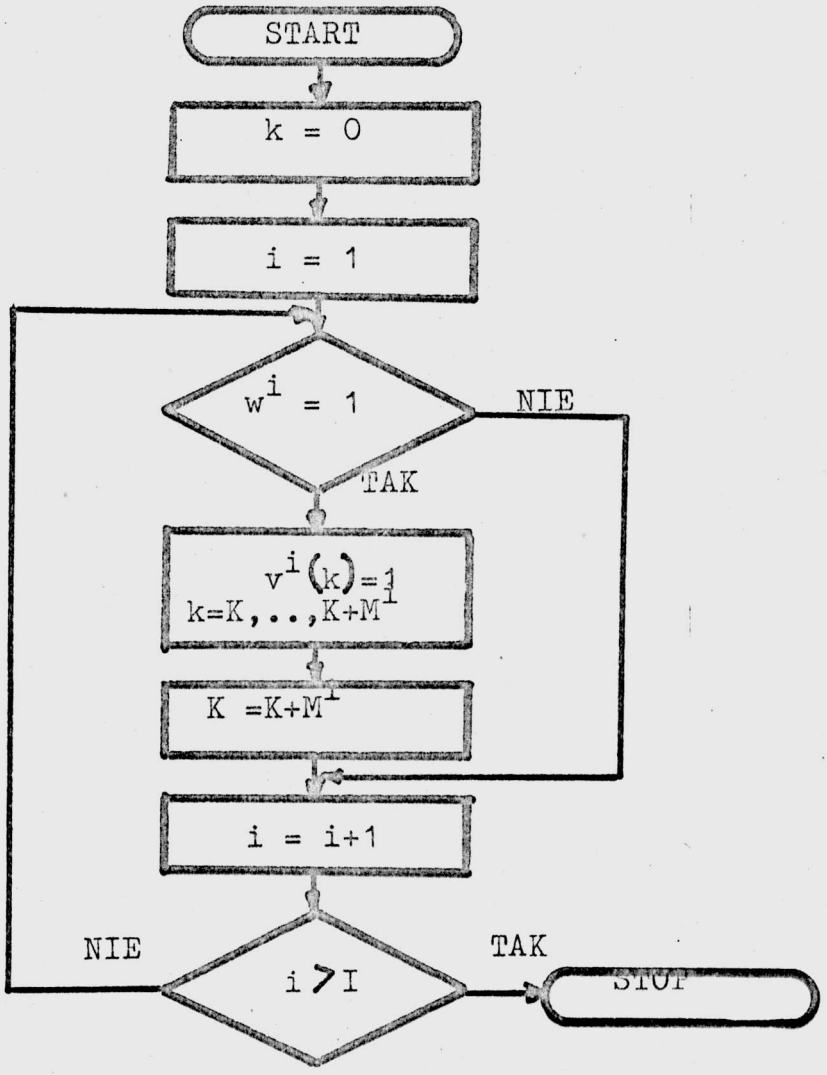
$$d^1 \leq d^2 \leq \ldots \leq d^I$$

wszystkie powyższe rozważania zachowują swoją moc. Jedynie jeżeli dla pewnego j, $d^j = d^{j+1}$, j-te ograniczenie jest słabsze niż j+1-sze , i może być pominięte. W przypadku, gdy nie wyrugujemy nieistotnych ograniczeń, wszystkie rozważania są poprawne ale niepotrzebnie rośnie wymiar zadania optymalizacji.

Realizacja otrzymanego algorytmu w KSS

Algorytm procedury szeregowania SO realizującej optymalny rozdział zasobów jest bardzo podobny do algorytmu zamieszczonego na rysunku 2.1.
Jedynie zadanie optymalizacji powstające na każdym kroku sterowania jest nieco bardziej skomplikowane. Natomiast algorytm szeregowania programów dla których $w^i=1$ /rys.4.1/ ma bardzo prostą realizację komputerową.

należy liczby $w^i \cdot d^i \neq 0$ traktować jako prorytety /im mniejsza liczba tym większa ważność programu/, programy dla których $w^i \cdot d^i = 0$ nie kandydują do uruchomienia.



Rys. 4.1. Algorytm szeregowania programów

PRZYKŁAD

Niech w zadaniu rozpatrywanym w tym rozdziale

$I=10,\ M^1=2, M^2=1, M^3=4, M^4=7, M^5=5, M^6=10, M^7=12, M^8=15, M^9=10, M^{10}=10$

$d^1=3,\ d^2=5, d^3=10, d^4=15, d^5=18, d^6=20, d^7=25, d^8=30, d^9=35, d^{10}=35;$

a funkcja celu /4.10/ ma postać

$$Q= -3w^1-5w^2-7w^3-2w^4-1w^5-4w^6-3w^7-8w^8-2w^9-9w^{10}$$

Wówczas ograniczenia /4.11/ przyjmą postać

$$2w^1 \leq 3$$
$$2w^1+1w^2 \leq 5$$
$$2w^1+1w^2+4w^3 \leq 10$$
$$2w^1+1w^2+4w^3+7w^4 \leq 15$$
$$2w^1+1w^2+4w^3+7w^4+5w^5 \leq 18$$
$$2w^1+1w^2+4w^3+7w^4+5w^5+10w^6 \leq 20$$
$$2w^1+1w^2+4w^3+7w^4+5w^5+10w^6+12w^7 \leq 25$$
$$2w^1+1w^2+4w^3+7w^4+5w^5+10w^6+12w^7+15w^8 \leq 30$$
$$2w^1+1w^2+4w^3+7w^4+5w^5+10w^6+12w^7+15w^8+10w^9 \leq 35$$
$$2w^1+1w^2+4w^3+7w^4+5w^5+10w^6+12w^7+15w^8+10w^9+10w^{10} \leq 35$$

w^i binarne , $i=1,2,..,10.$

Otrzymane zadanie optymalizacji zostało rozwiązane i otrzymano następujące wyniki:

$$\overset{*}{w}{}^1=1, \overset{*}{w}{}^2=1, \overset{*}{w}{}^3=1, \overset{*}{w}{}^4=0, \overset{*}{w}{}^5=0, \overset{*}{w}{}^6=0, \overset{*}{w}{}^7=0, \overset{*}{w}{}^8=1, \overset{*}{w}{}^9=0, \overset{*}{w}{}^{10}=1.$$

Wartość funkcji celu $\overset{*}{Q} = -32$

Program w języku FORTRAN rozwiązujący ten problem zamieszczono w Dodatku do niniejszej pracy [56]. ■

4.2.2. Rozdział wielu zasobów.

Omówimy teraz zadanie rozdziału wielu zasobów w KSS obiektami statycznymi w przypadku, gdy w każdym takcie sterowania każdy program ma inną linię krytyczną. Podobnie jak w 4.2.1. rozpatrywać będziemy tylko jeden krok sterowania. Zadanie rozdziału zasobów w tym przypadku sformułowane jest w sposób następujący:

znaleźć liczby $v^{*i}(k)$, $i=1,2,..,I$; $0 \leqslant k < K=\max_i\{d^i\}$
minimalizujące Q

$$Q = \sum_{i=1}^{I} a^i \varphi^i\left(f^i\left(\gamma^i\left(x^i(a^i), z^i\right), z^i\right)\right) \qquad (4.12)$$

przy ograniczeniach

$$\forall_k \sum_{i=1}^{I} v^i(k) \cdot a^i(k) \leqslant c \qquad (4.13)$$

$$v^i(k) = 0 \quad , \quad k \geqslant a^i$$

$$x^i(k+1) = x^i(k) + v^i(k)$$

$$x^i(0) = 0$$

$v^i(k)$ binarne

Do rozwiązania tego zadania sterowania zastosujemy metodę programowania dynamicznego, a na każdym jego etapie algorytm oparty na metodzie podziału i ograniczeń.

Założymy, że programy ponumerowane są w ten sposób, że:

$$d^1 \leqslant d^2 \leqslant \ldots \leqslant d^I$$

Niech B_k oznacza funkcję Bellmana na k-tym etapie obliczeń, $k=K-1, K-2, \ldots, 1, 0$

$$B_k(x(k)) = \min_{v \in V_k} B_{k+1}(x(k)+v(k)) + \sum_{i \in \gamma_k} \varphi^i\left(f^i\left(\gamma^i\left(x^i(k), z^i\right), z^i\right)\right)$$

$$B_K(x) = 0$$

gdzie:

$$x(k) = \begin{bmatrix} x^1(k) \\ \vdots \\ x^I(k) \end{bmatrix}, \quad v(k) = \begin{bmatrix} v^1(k) \\ \vdots \\ v^I(k) \end{bmatrix} \qquad (4.14)$$

$$\gamma_k = \{i : k = a^i\}, \quad V_k = \left\{v(k) : \sum_{i=1}^{I} a^i(k) \cdot v^i(k) \leqslant c\right\}$$

γ_k jest to zbiór tych programów których termin zakończenia upływa w chwili k.

Oznaczmy funkcję minimalizowaną na każdym etapie procedury programowania dynamicznego przez $\tilde{B}_k(v(k))$. Do minimalizacji $\tilde{B}_k$ wykorzystamy algorytm oparty na metodzie podziału i ograniczeń. Metoda ta została pobieżnie omówiona w rozdz. 3.1.2. dalsze szczegóły można znaleźć w wielu publikacjach, min. [29,46]. Poniżej podane zostaną jedynie podstawowe elementy tej metody tzn. zasady wyliczania oszacowań i wyboru zmiennej do podziału.

Zbiór rozwiązań dopuszczalnych w wierzchołku h_1 rozwiązań określony jest w następujący sposób:

$$V_\lambda = \{v(k) : v^i(k) = 1,\ i \in I_\lambda^+,\ v^i(k) = 0,\ i \in \bar{I}_\lambda,\ \sum_{i \in \bar{I}_k} v^i(k) a^i(k) \le c - \sum_{i \in I_k^+} a^i(k)\}$$

I_1, I_1^+, I_1^-, $\bar{I}_1$ określone zostały w 3.1.2.

Wszystkie rozważania będą dotyczyły jednego, ustalonego etapu programowania dynamicznego.

Wyliczanie oszacowań

Podamy teraz metodę szacowania wartości minimum $\tilde{B}_k(v(k))$ na zbiorze V_1 tzn.

$$\underline{z}_1 \le z_1 = \min_{v(k) \in V_1} \tilde{B}_k(v(k)) \le \bar{z}_1$$

Jeżeli

$$c - \sum_{i \in I_1^+} a^i(k) = s < 0 \qquad /4.15/$$

wówczas $V_\lambda = \phi$ i $\underline{z}_k = +\infty$

Jeżeli $0 \le s$ i $\forall i \in \bar{I}_k\ a^i(k) > s$ /4.16./

wówczas $V_\lambda = \{\underline{v}\}$

$$\underline{v} = \begin{bmatrix} v^1 \\ \vdots \\ v^I \end{bmatrix} \quad \begin{array}{ll} v^i = 0 & i \notin I_k^+ \\ v^i = 1 & i \in I_k^+ \end{array} \qquad \underline{z}_k = \tilde{B}_k(\underline{v})$$

w przeciwnym przypadku

$$\underline{z}_k = \tilde{B}_k(\underline{v})$$

gdzie $\underline{v} = \begin{bmatrix} v^1 \\ \vdots \\ v^I \end{bmatrix}$ $\quad \underline{v}^i = 0\ i \in I_k^-,\ \underline{v}^i = 1\ i \notin I_k^-$

Jak łatwo zauważyć zarówno $\tilde{B}_K$ jak i $\tilde{B}_k$, $k = K-1, K-2, \ldots$ są funkcjami nierosnącymi każdego z argumentów $v^i(k)$.

Jako oszacowanie górne $\bar{z}_1$ przyjmiemy wartość funkcji w dowolnym punkcie $\bar{v} \in V_1$

np.: $\bar{v}^i = 1$, $i \in I_k^+$; $\bar{v}^i = 0$, $i \notin I_k^+$

Wybór zmiennej do podziału

Sposób wyboru zmiennej do podziału jest podobny do opisanego w 3.2.1.

Dla każdego $i \in \bar{I}_1 i$ takiego że $a^i(k) \leq s$ wyliczamy Δ_1^i

$$\Delta_\ell^i = \tilde{B}_k(v^{(i)}) - \tilde{B}_k(\tilde{v})$$

gdzie

$$v^{(i)} = \begin{bmatrix} v^{(i)1} \\ \vdots \\ v^{(i)I} \end{bmatrix}, \quad \tilde{v} = \begin{bmatrix} \tilde{v}^1 \\ \vdots \\ \tilde{v}^I \end{bmatrix} \quad v^{(i)} \in V_\ell, \ \tilde{v} \in V_\ell$$

$$v^{(i)j} = 0 \quad j \in \bar{I}_k, \ j \neq i \qquad v^{(i)i} = 1$$

$$\tilde{v}^j = 0 \quad j \in \bar{I}_k$$

Do podziału wybierzemy tę zmienną i^0, dla której

$$\Delta^{i^0} = \min_{i \in \bar{I}_k} \Delta^i$$

Poniżej zapisany został algorytm rozwiązywania zadania minimalizacji B_k oparty na metodzie podziału i ograniczeń.

Krok 1 - początkowy. W wierzchołku h_0: $I_0 = \Phi$, $\underline{z}_0 = -\infty$ $\bar{z}_0 = +\infty$, $l=1$. Przejdź do kroku 2

Krok 2 - obliczanie oszacowań. Wylicz oszacowanie w h_ℓ wg /4.8/ lub /4.8.'/ Podstaw $\bar{z} = \min\{\bar{z}_0, \bar{z}_\ell\}$

Krok 3 - zamykanie wierzchołków. Jeżeli zachodzi /4.8/ lub /4.8 / lub/4.8/,/4.8ub $\bar{z}_\ell = \underline{z}_\ell$, to zamknij h_ℓ i przejdź do kroku 4. W przeciwnym razie przejdź do kroku 5.

Krok 4 - cofanie się. Jeżeli nie ma wierzchołków aktywnych przejdź do kroku 6. W przeciwnym przypadku wybierz gałąź prowadzącą do ostatnio otrzymanego wierzchołka aktywnego i przejdź do kroku 2.

Krok 5 - podział i gałęzienie. Dokonaj rozbicia wg /2.8/ wybierając zmienne ω^{i^o} zgodnie z /4.9/. Wybierz gałąź $\omega^{i^o} = 1$. Przejdź do kroku 3.

Krok 6 - końcowy. Jeżeli $\bar{z}_o = +\infty$ to zadanie nie ma rozwiązania dopuszczalnego; jeżeli $\bar{z}_o < \infty$ to rozwiązanie dopuszczalne, które daje $\bar{z}_o$ jest rozwiązaniem optymalnym.

5. SPOSÒB REALIZACJI ALGORYTMÒW W KOMPUTEROWYCH SYSTEMACH STERUJĄCYCH.

W rozdziale tym podane zostaną uwagi dotyczące sposobu i możliwości realizacji opisanych w niniejszej prady algorytmów optymalnego rozdziału zasobów.

5.1. Rozwiązywanie zadania optymalizacji na bieżąco

Jak już niejednokrotnie podkreślano, postawione w pracy zadania optymalizacji dyskretnej są skomplikowane pod względem obliczeniowym. Dlatego też, aby móc ocenić możliwość ich stosowania w SO czasu rzeczywistego dokonamy próby oceny złożoności obliczeniowej otrzymanych algorytmów. Dla jej oceny wykorzystać można porównanie postawionych zadań z zadaniem szeregowania programów przed liniami krytycznymi czy zadaniem minimalizacji czasu wykonania zbioru programów. Jak pokazano, zadania te stanowią część algorytmów optymalnego rozdziału zasobów. W przypadku, gdy wszystkie programy mogą być wykonane w zadanym czasie, rozpatrywane w pracy problemy sprowadzają się do zadania szeregowania przed liniami krytycznymi. Efektywne algorytmy do rozwiązywania tych zadań istnieją jedynie w przypadku rozdziału jednego zasobu - procesora [6,8,9] . Zadania szeregowania z uwzględnieniem dotatkowych zasobów są o wiele trudniejsze i większość algorytmów jest NP - zupełna: [6,8,9] . W zadaniach rozpatrywanych w niniejszej pracy dodatkowym utrudnieniem jest zadanie optymalizacji całkowitoliczbowej związane z wyborem programów do wykonania. I tak, np. zagadnienie załadunku rozpatrywane w 2.1.2. jest problemem NP - zupełnym, acakolwiek istniejące algorytmy są stosunkowo efektywne.

Z powyższych uwag wynika, że zadania optymalizacji powstające przy szukaniu najlepszego rozdziału zasobów będą mogły być rozwiązywane na bieżąco jedynie w pewnych przypadkach. Na przykład zadanie P1 może być rozwiązywane w czasie rzeczywistym, Natomiast zadanie P1 badane w 2.2. może być rozwiązywane na bieżąco jedynie w przypadku niewielkiej liczby programów. Do jego rozwiązania można również stosować algorytm załadunku po dokonaniu redukcji ograniczeń, jak opisano to w 3.2.1.

Natomiast nie wydaje się możliwe rozwiązywanie na bieżąco zadań optymalizacji powstających podczas rozdziału zasobów w KSS obiektami dynamicznymi. Problem ten omówiony zostanie dokładniej w rozdziałach 5.2. i 5.3.

Na zakończenie, dodać należy, że procedury rozdziału zasobów SC rozwiązujące zadanie optymalizacji na bieżąco są najbardziej uniwersalne. Informacje o czasie obliczeń poszczególnych programów czy liczbie dostępnych zasobów mogą być podane na początku każdego kroku sterowania. Procedury takie łatwo przystosowują się do zmian zachodzących w otoczeniu, kosztem stosunkowo długiego czasu ich realizacji.

Osobnego omówienia wymaga założenie uczynione na początku pracy o możliwości pominięcia czasu zużywanego przez SO na podejmowanie decyzji. Jest to oczywiście pewna stylizacja rzeczywistości. Jednak w przypadku rozpatrywanym w tej pracy wydaje się być do przyjęcia. Wynika to z następującego faktu: decyzje SO podejmowane są na podstawie zmierzonych parametrów obiektów sterowania na początku każdego kroku sterowania. Czas zużyty na podjęcie decyzji nie może być wykorzystany przez żaden z programów, którego decyzje te dotyczą. Zatem we wszystkich rozważaniach można przyjąć, że T oznacza czas dostępny programom użytkowym i nie obejmuje czasu zużytego przez SO.

5.2. Realizacja algorytmów rozdziału zasobów w układzie zamkniętym.

Wprowadzamy terminologię zaczerpniętą z teorii sterowania, rozumiejąc przez układ sterowania w układzie zamkniętym, układ pracujący, że sprzężeniem zwrotnym, mierzący stan /parametr obiektu i na tej podstawie wydający decyzje.
Rozdział zasobów w układzie otwartym polega na realizacji wcześniej wyliczonych sterowań.

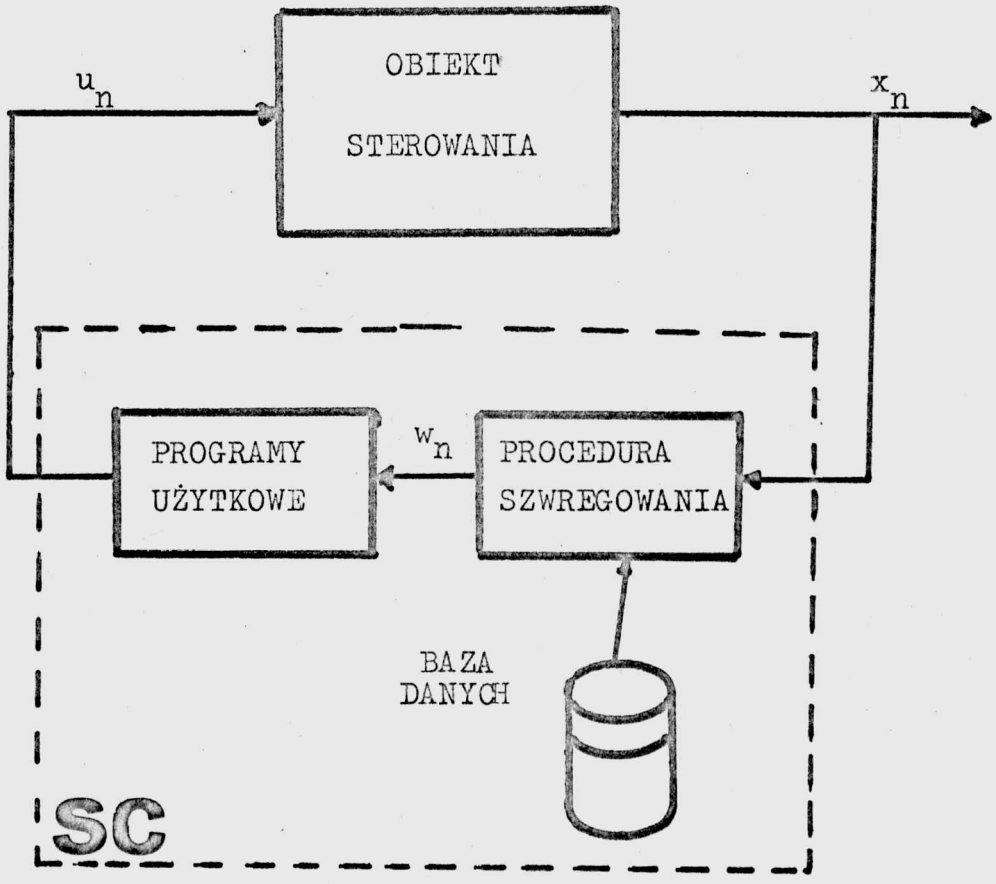
Alternatywą do rozwiązywania zadania optymalizacji na bieżąco jest wielokrotne rozwiązanie go wcześniej dla ustalonych wartości parametrów obiektu, a następnie wybór odpowiedniego wariantu sterowania w zależności od aktualnej wartości parametru.

Postępowanie takie ma szereg zalet układu pracującego ze sprzężeniem zwrotnym /tzw. mniejsza wrażliwość na wszelkie niedokładności pomiarów, opisu obiektu itp/, a również jest stosunkowo prosta w realizacji. Do wad zaliczyć można konieczność zapamiętania stosunkowo dużej ilości informacji o rozwiązaniach. Tym bardziej, że nie można uzyskać analitycznej zależności pomiędzy decyzjami SO a wartością stanu/zaburzeń.

Inną wadą jest to, że tak skonstruowana procedura jest mniej uniwersalna: reaguje na zmiany zachodzące w otoczeniu, ale nie może reagować na zmiany zachodzące w SC /liczbę dostępnych zasobów, czasy trwania programów, itp/.
Co prawda uwzględnienie zmian tych wielkości jest możliwe, ale wówczas liczba informacji koniecznych do zapamiętania rośnie bardzo szybko.

W celu uzyskania algorytmów rozdziału zasobów w układzie zamkniętym należy zadania optymalizacji postawione w rozdz. 2 rozwiązać wielokrotnie dla różnych wartości zakłóceń. Analogiczną metodą stosuje się przy rozwiązywaniu zadań optymalizacji na każdym etapie programowania dynamicznego. Czas potrzebny na rozwiązanie tego zadania jest nieistotny, ponieważ jest ono rozwiązywane przed rozpoczęciem zadania sterowania. Odpowiada to zadaniu projektowania regulatora. Tak otrzymana rozwiązania powinny być następnie zapisane w odpowiedniej bazie danych, z której korzysta SO podczas szeregowania programów.

Na rysunku 5.1. przedstawiono schematycznie realizację algorytmu rozdziału zasobów w układzie zamkniętym.



Rys. 5.1. Schemat realizacji algorytmu rozdziału zasobów w układzie zamkniętym.

5.3. Sterowanie rozdziałem zasobów w układzie otwartym

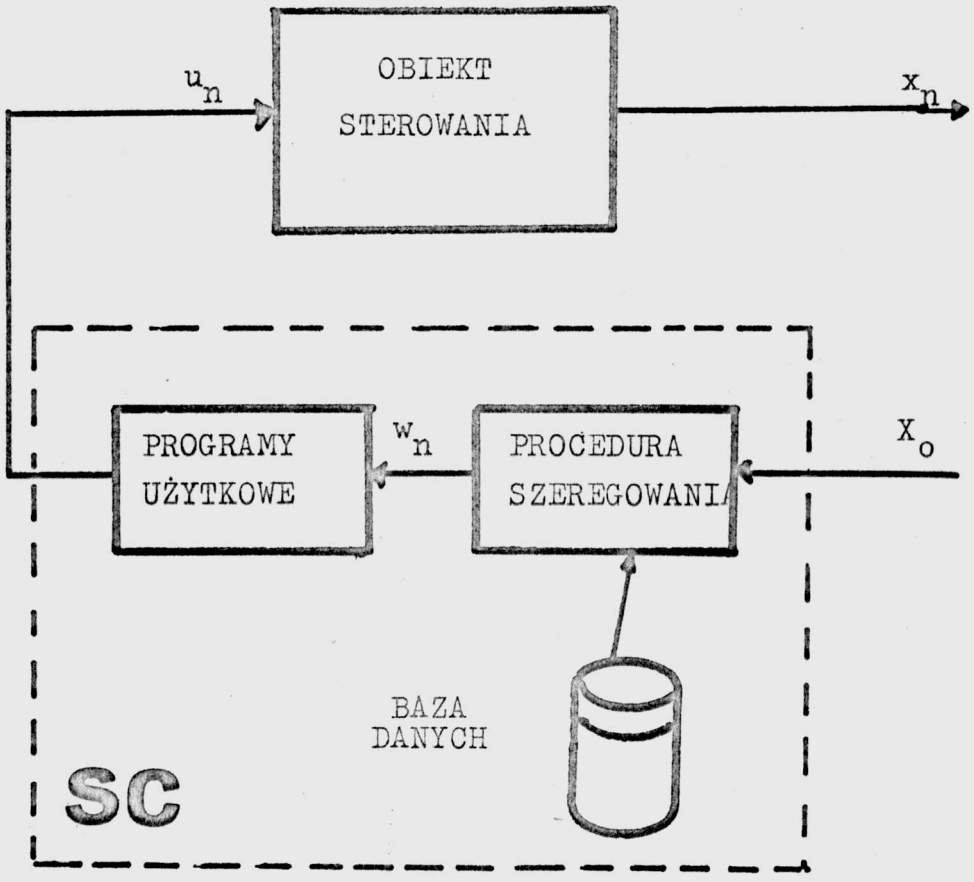
Tego rodzaju procedury rozdziału zasobów stosowane być mogą jedynie w KSS obiektami dynamicznymi. W przypadku obiektów statycznych założenie o nieznajomości zaburzeń i możliwości ich pomiaru powoduje, że decyzje podejmowane być muszą w układzie zamkniętym. Procedury szeregowania pracujące w układzie otwartym w porównaniu z innymi są jeszcze mniej uniwersalne, wykazują większą wrażliwość na niedokładności opisu, zależne są od stanu początkowego obiektu sterowania.

Do ich zalet zaliczyć można znaczną prostotę realizacji i stosunkowo niewielką ilość informacji binarnych niezbędnych do przechowania.

W przypadku realizacji procedury szeregowania w układzie otwartym wektory w_n /lub $v_n(k)$ / mogą być zapisane w pamięci stałej SC lub w odpowiednio zorganizowanej bazie danych. Innym możliwym podejściem jest budowa osobnego urządzenia /automatu/, którego wyjście przyjmuje wartości wyliczonych decyzji.

Taka realizacja jest szczególnie szybka, ale możliwa do zastosowania jedynie tam, gdzie proces technologiczny powtarza się cyklicznie w identycznych warunkach.

Na rysunku 5.2. przedstawiono schematycznie realizację procedury szeregowania w układzie zamkniętym.



Rys. 5.2. Realizacja algorytmu rozdziału zasobów w układzie otwartym.

6. ILUSTRACJA ZADANIA ROZDZIAŁU ZASOBÓW NA PRZYKŁADZIE KSS KONWERSJĄ SO_2.

W rozdziale podana zostanie ilustracja metod rozdziału zasobów zaproponowanych w pracy, w KSS realizującym pewne zadanie sterowania wybranym fragmentem procesu produkcji kwasu siarkowego.

Opis procesu technologicznego [60]

Proces produkcji kwasu siarkowego metodą kontaktową w Z. Ch. POLICE przedstawić można w sposób następujący: płynna siarka ze zbiorników siarki tłoczona jest poprzez filtry do pieca, gdzie jest spalana w nadmiarze powietrza. Powietrze do spalania siarki osuszane jest przy pomocy kwasu siarkowego i ogrzewane w wymiennikach ciepła gazami reakcyjnymi. Spaliny z pieca o temperaturze ok. 1000°C rozdzielane są na dwie części; część schładzana jest w wymienniku ciepła przekazując ciepło parze wodnej, wykorzystywanej następnie do celów technologicznych i energetycznych. Schładzane gazy mieszane są w odpowiednim stosunku z gorącymi i podawane do aparatu kontaktowego. Składa się on z czterech zamkniętych przestrzeni tzw. półek, ułożonych jedna nad drugą.

Gazy wychodzące z półki położonej wyżej przechodzą na następną położoną niżej poprzez kolejny wymiennik ciepła.

W aparacie kontaktowym zachodzi konwersja dwutlenku siarki na trójtlenek. Katalizatorem jest masa wanadowa

$$SO_2 + \frac{1}{2} O_2 \overset{V_2O_5}{\rightleftharpoons} SO_3 \qquad \text{KJ/mol}$$

Gaz po czwartej półce chłodzi się przeponowo do temp. ok. 200°C w podgrzewaczu wody. Następnie wchodzi od dołu do wieży absorbcyjnej, zraszanej kwasem siarkowym. Trójtlenek siarki absorbuje się w kwasie, a gazy odpadowe wypływają z wieży do komina. Zagęszczony kwas siarkowy jest rozcieńczony wodą i odprowadzany do zbiornika kwasu. Wydajność roczna jednego ciągu wynosi ok. 210 tys.ton H_2SO_4 na rok. W Z.Ch.POLICE jest sześć takich ciągów; na rys.6.1. przedstawiono uproszczony schemat jednej nitki tech-

nologicznej.

Model matematyczny procesu konwersji [26,61,62,63,27]

Podamy bardzo uproszczony matematyczny opis procesu konwersji zachodzącego na jednej półce.

Zmiany stężenia SO_2 w gazach reakcyjnych, w stanie ustalonym mogą być opisane następującą zależnością /podajemy opis jednej nitki/

$$\frac{dc_A^i}{dx} = -\gamma^i r_A^i \left(t_g(x)\right) \tag{6.1}$$

gdzie: $c_A^i(x)$ stężenie SO_2 w gazie w pkt x,

x współrzędna przestrzenna,

$t_g^i(x)$ temperatura gazu w pkt x,

$r_A^i(x)$ oznacza szybkość reakcji,

z warunkiem początkowym $c_A^i(0) = c_A^{iw}$

Zmiany temparatury gazu można opisać w stanie ustalonym następującym równaniem różniczkowym

$$\frac{dt_g^i}{dx} = \alpha^i \left(t_k^i(x) - t_g^i(x) \right) - \bar{Q}\left(t_g^i(x)\right) \tag{6.2}$$

gdzie: $t_k^i(x)$ - temperatura katalizatora

Q - ciepło reakcji

z warunkiem początkowym $t_g^i(x) = t_g^{iw}$

Dla potrzeb pracy przyjęto, że tempo reakcji r_A^i z wystarczającą dokładnością może być aproksymowane formą kwadratową

$$r_A^i \left(t_g(x)\right) = A^i \left(t_g^i(x)\right)^2 + B^i \left(t_g^i(x)\right) + C^i \tag{6.3}$$

Przyjęto również, że temperatura katalizatora jest niezależna od temperatury gazu i określona krzywą przedstawioną na rys. 6.2.

Wyrażenie na ciepło reakcji $\bar{Q}^i$ zaczerpnięto z [6.1]:

$$\bar{Q}\left(t_g^i\right) = 22034.3 + 5.618 \cdot t_g^i - 10.4575\ E\text{-}3 \cdot t_g^{i2} + 6.4212\ E\text{-}6 \cdot t_g^{i3} - 1.648\ E - 9\ t_g^{i4}$$

Parametry A^i, B^i, C^i, α^i wyestymowano metodą najmniejszych kwadratów na podstawie pomiarów [62] i tak np.:

$$A^1 = -1.668655\ E - 05$$
$$B^1 = 0.016237366$$
$$C^1 = 2.6294968\ E - 06$$
$$\alpha^1 \approx 743$$

W procesie konwersji zależy nam na tym aby jak najwięcej SO_2 przereagowało w SO_3; w przyjętej konwencji oznaczeń odpowiada temu minimalizacja $c_A^i(1)$. Ponieważ

$$c_A^i(x) = c_A^{iw} - \gamma^i \int_0^x r_A^i\left(t_g^i(x)\right) dx \qquad (6.4)$$

zadanie to jest równoważne zadaniu maksymalizacji $\int r_A^i\left(t_g^i(x)\right) dx$

Przebieg temperatury gazu, a co za tym idzie tempo reakcji zależy od temperatury gazów wlotowych, ta zaś może być zmieniona poprzez ustalenie ilości gazów omijających przegrzewacz pary. Jeżeli u^i oznacza ułamek gazów omijających wymiennik ciepła temperatura gazów wpływających do aparatu kontaktowego określona jest następującą zależnością:

$$t_g^{iw} = u^i \cdot \bar{t}^i + (1-u^i) \cdot \bar{\bar{t}}^i \qquad (6.5)$$

$\bar{t}^i$ - temperatura gazów wychodzących z pieca,

$\bar{\bar{t}}^i$ - temperatura gazów po wymienniku

$0 \leqslant u^i \leqslant 1$, $i=1,2,\ldots,6$

Temperatura gazów wychodzących z pieca traktowana jest jako zaburzenie /zmienia się z powodu zmian parametrów siarki, temperatury powietrza wchodzącego do pieca/. Temperatura gazów wychodzących z wymiennika ciepła również zmienia się w zależności od zmian parametrów wody wchodzącej do wymiennika i zmian zapotrzebowania na parę.

Na podstawie obserwacji danych pomiarowych można zauważyć, że parametry procesu pozostają niezmienne na odcinku ok. 15-20 minut. Przyjmiemy, że sterowania muszą być wydawane z okresem 15 minutowym /decyzja ta zawsze musi być uzgadniana z technologiem procesu/.

Tak więc zadanie sterowania jednym ciągiem technologicznym może być przedstawiona w następujący sposób:

znaleźć ciąg liczb u_n^{*i}, $n=0,1,\ldots,N-1$; $i=1,2,..,6$

minimalizujące Q^i

$$Q^i = \int_0^{\hat{}} r_A^i\left(t_g^i(x)\right)dx \qquad (6.6)$$

gdzie

$$\frac{dt_{gn}^i}{dx} = \alpha^i\left(t_{kn}^i(x) - t_{gn}^i(x)\right) - Q\left(t_{gn}^i(x)\right)$$

z warunkiem początkowym

$$t_{gn}^i(0) = u_n^i\,\bar{t}_n^i + \left(1-u_n^i\right)\bar{\bar{t}}_n^i$$

W wyniku rozwiązania powyższego zadania otrzymamy następujący algorytm wyliczenia sterowań:

$$u_n^{*i} = \varphi^i\left(\bar{t}_n^i, \bar{\bar{t}}_n^i\right) \quad , \quad i=1,2,..,6.$$

Funkcja φ^i jest iteracyjną metodą minimalizacji funkcji jednej zmiennej.

Zadanie rozdziału zasobów

W tak opisanym systemie sterowania przyjąć można, że każdy z sześciu programów ma jednakowe żądania zasobowe i czas obliczeń.

Przyjmując notację podaną w rozdz. 2.1. zadanie rozdziału zasobów może być zapisane w następujący sposób:

znaleźć liczby w_n^{*i}, $n=0,1,\ldots,N-1$; $i=1,2,..,6$

mininalizujące Q

$$Q = \sum_{i=1}^{6} Q^i\left(u^i\right)$$

gdzie Q^i określona jest zależnością /6.6/.

Sterowania na obiekty wydawane są według następującej reguły:

$$u_n^i = w_n^i \cdot \psi_n^i\left(\bar{t}_n^i, \bar{\bar{t}}_n^i\right) + \left(1-w_n^i\right)\cdot \tilde{u}^i$$

przy ograniczeniach:

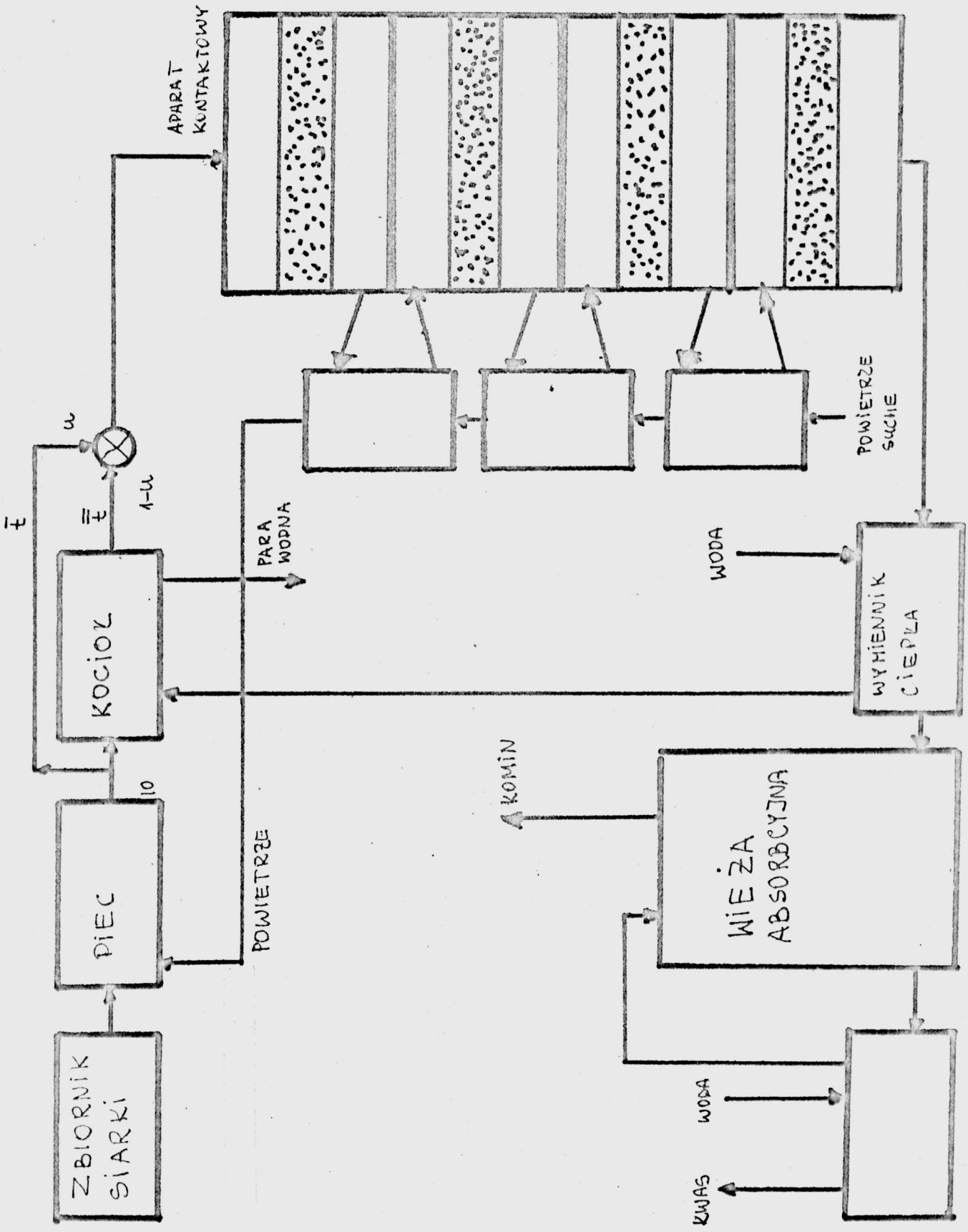
$$\sum_{i=1}^{6} w_n^i \cdot M_n^i \leq T \quad , \quad n=0,1,\ldots,N-1$$

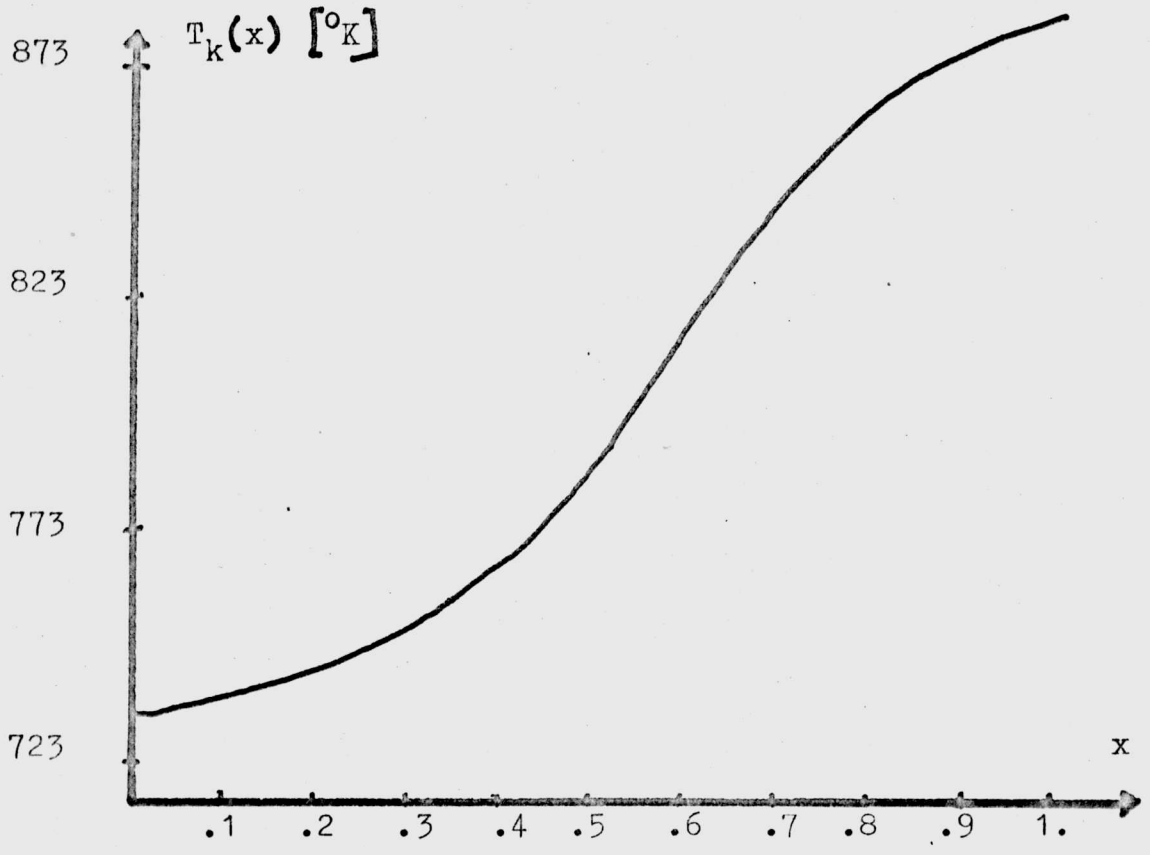
Założyliśmy, że sterowania realizowane są przez jednoprocesorowy system cyfrowy.

Sterowania zastępcze $\tilde{u}^i$ przyjęto rozwiązując zadanie optymalizacji /6.6/ dla znamionowych wartości temperatur $\bar{t}_n^i$, $\bar{\bar{t}}_n^i$; i tak dla jednego z obiektów i $\bar{t}_n^1 = 1273^o$ K, $\bar{\bar{t}}_n^1 = 703^o$ K, $\tilde{u}^1 = .05$.

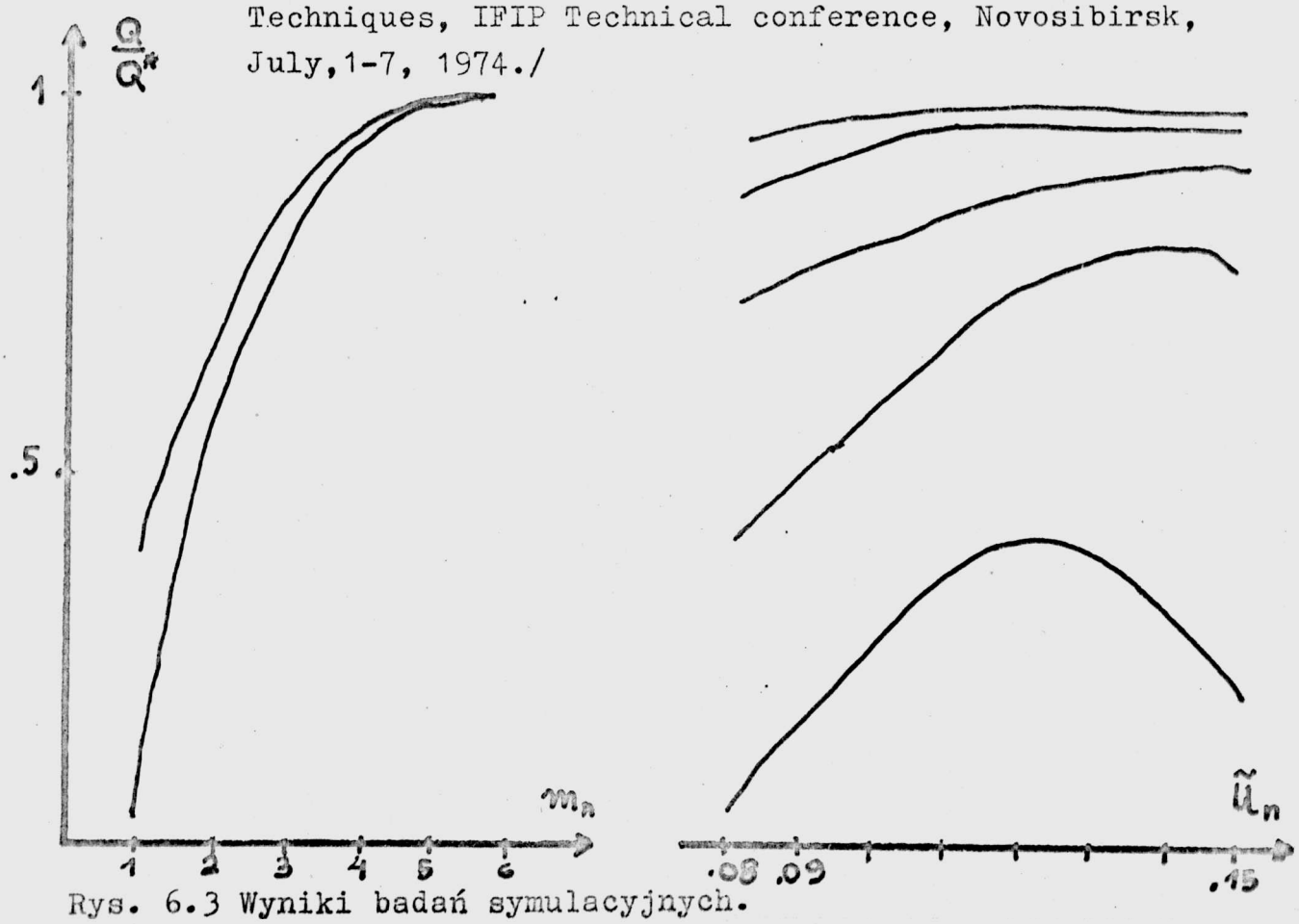
Badania symulacyjne przeprowadzono przy założeniu, że opis wszystkich ciągów technologicznych jest jednakowy oraz jednakowa jest wartość sterowania zastępczego dla wszystkich obiektów. Temperatury $\bar{t}_n^i$ i $\bar{\bar{t}}_n^i$ losowano zgodnie z rozkładem jednostajnym na odcinkach $[1233,1353]$ ^{o}K i $[573,723]$ ^{o}K odpowiednio. Wyniki symulacji /zależność wartości wskaźnika jakości od liczby uruchamianych programów m_n /por. 2.1.1/ oraz wartości sterowania zastępczego/przedstawiono na wykresach / rys. 6.3/

Rys. 6.1 Uproszczony schemat technologiczny produkcji H_2SO_4





Rys. 6.2 Temperatura katalizatora / zaczerpnęto z pracy: L. Le Letty, Studies in modelling and identification of distributed parameter systems, w: Optimization Techniques, IFIP Technical conference, Novosibirsk, July, 1-7, 1974./



Rys. 6.3 Wyniki badań symulacyjnych.

7. ZAKOŃCZENIE

W pracy postawiono i rozwiązano zadanie optymalnego rozdziału zasobów w wybranej klasie Komputerowych Systemów Sterujących procesami technologicznymi.

Jak dotąd brak było w literaturze podobnych badań, ujmujących wpływ algorytmów SO na jakość realizowanego sterowania.

Otrzymane w pracy algorytmy rozdziału jednego zasobu w KSS obiektami statycznymi są bardzo proste co pozwala na ich realizację w Systemach Operacyjnych.

Algorytm rozdziału wielu zasobów zaproponowany w rozdziale 2.2. służyć może do rozwiązywania zadań szeregowania programów przed liniami krytycznymi z uwzględnieniem dodatkowych zasobów. Może być on wykorzystany do konstruowania procedur rozdziału zasobów, pod warunkiem, że zastosowana zostanie odpowiednio efektywna metoda rozwiązywania otrzymanego zadania optymalizacji.

W przypadku KSS obiektami dynamicznymi otrzymane zadanie optymalizacji jest znacznie trudniejsze i w zasadzie nie może być rozwiązywane na bieżąco.
Pewnym rozwiązaniem może być, zaczerpnięta z metod teorii sterowania idea analitycznego /numerycznego/ konstruowania regulatora.

Takie podejście pozwala na realizację optymalnych algorytmów w przypadku bardzo skomplikowanych zadań. Daje ono również podstawy do konstruowania algorytmów najlepszych w pewnej klasie /określonej np. parametrycznie/.

Znaczenie uzyskanych w pracy rezultatów wzmacnia fakt zaproponowania metod postępowania przy konstruowaniu algorytmów w pewnych ogólniejszych i bliższych rzeczywistości przypadkach.

W dalszych badaniach warto rozszerzyć wyniki uzyskane w tej pracy na Komputerowe Systemy Sterujące ogólniejszymi klasami obiektów sterowania. Należy zbadać również własności optymalnego rozdziału zasobów w przypadku innych niż rozpatrywanie zadań sterowania.

Na koniec warto zauważyć, że możliwość praktycznego stosowania algorytmów optymalizacji rozdziału zasobów ograniczona jest przez efektywność stosowanych metod optymalizacji.

Dlatego też, w dalszych badaniach należy zająć się poszukiwaniem najefektywniejszych metod rozwiązywania postawionych zadań, szukać efektywnych algorytmów suboptymalnych lub ograniczyć się do poszukiwań algorytmów rozdziału zasobów w pewnej klasie.

LITERATURA

[1.] Aven, O.I., Kogan, Ja.A. Upravlenie vycislitel'nym processom v EVM. /Algoritmy i modeli/. Moskva, 1978.

[2.] Awo, A.V., Hopcroft, J.E., Ullman, J.D. The design and analysis of computer algorithms. Addison - Wesley, 1976.

[3.] Blackman, M. The design of real-time applications. Willey, London, 1975.

[4.] Błażewicz, J. Deadline scheduling of tasks. A survey. Foundations of Control Engineering, Vol 1. No 4, 1976.

[5.] Błażewicz, J. On scheduling to meet deadlines and resource constrains. Proc. of the 11th Yugoslav International Symposium on Information Processing - Informatica 76,Bled, 1976.

[6.] Błażewicz, J., Cellary, W., Węglarz, J. Słowiński, R. Deterministyczne problemy szeregowania zadań na równoległych procesorach. Cz. I. Zbiory zadań niezależnych. Cz. II. Zbiory zadań zależnych. Podstawy Sterowania, Tom 6, Z. 1, Z. 2, 1976.

[7.] Błażewicz, J., Cellary, W., Węglarz, J. A strategy for scheduling splittable tasks to reduce schedule lenght. Acta Cybernetica, T.3, Fasc. 2, 1977.

[8.] Błażewicz, J., Węglarz, J. Deterministyczne problemy szeregowania zadań na procesorach systemów komputerowych z uwzględnieniem dodatkowych zasobów. Archiwum Automatyki i Telemechaniki, T. XXIII, Z. 4, 1978.

[9.] Błażewicz J. Złożoność obliczeniowa algorytmów i problemów szeregowania zadań. Politechnika Poznańska, Rozprawy 104, 1979.

[10.] Bołtiański, W.G. Sterowanie optymalne układami dyskretnymi. WNT, 1978.

[11.] Borzemski, L., Buchalski, Z., Janiak, A., Mielcarek, J., Stanisz, A. System operacyjny dla zestawu ODRA 1325 SMA do sterowania eksperymentem w czasie rzeczywistym. Część III. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Raport 216. Wrocław, 1977.

[12] Borzemski, L., Buchalski, Z., Janiak, A., Kordecki, H., Myszka, W., Stanisz, A. System operacyjny dla zestawu ODRA 1325 SMA do sterowania eksperymentem badawczym w czasie rzeczywistym. Część IV. Instrukcje użytkowania. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Raport 281. Wrocław, 1978.

[13] Brinch Hansen, P. Podstawy systemów operacyjnych WNT, Warszawa, 1979.

[14] Bubnicki, Z. Problemy optymalnego sterowania kompleksami operacji. W: Materiały Konferencji Nowoczesne Problemy Automatyki i Informatyki Politechnika Śląska. Gliwice, 1973.

[15] Bubnicki, Z. Identyfikacja obiektów sterowania. PWN, Warszawa, 1974.

[16] Bubnicki, Z., Kordecki, H., Mielcarek, J., Myszka, W., Zdrzałka, Z. Założenia i koncepcja systemu operacyjnego dla zestawu ODRA 1325 + SMA do sterowania procesami chemicznymi. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Raport 209. Wrocław, 1977.

[17] Cellary, W. Task scheduling in systems with nonpreemptible resources. W: Measuring, modelling and evaluating computer system M.Beilner, E.Geleube /red/. North Holland Publ. Comp., 1977.

[18] Charnes, A., Raike, W.M. One - pass algorithm for some generalized network problems. Operations Research, Vol.14, No 5.

[19] Christofides, N. Graph theory. Academic Press, 1975.

[20] Coffman, E.G.jr./red/, Computer and job-schop scheduling theory, Wiley, 1976.

[21] Cohen, L.J. Operating systems. Analysis and design, Spartan Books.

[22] Colin, A.J.T. Wstęp do systemów operacyjnych. PWN, Warszawa, 1976.

[23] Conway, R.W., Maxwell, W.L, Miller, L.W, Theory of scheduling. Addison - Wesley, 1967.

[24] Cuttle, G., Robinson, P.B./red/ Programy zarządzające i systemy operadyjne. PWN, Warszawa, 1977.

[25] Dubielewicz, A. Metoda rozdziału funkcji systemu operacyjnego na realizacje sprzętowe i programowe /rozprawa doktorska/. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Raport Pre 47. Wrocław, 1979.

[26] Findeisen, W. Wielopoziomowe układy sterowania. PWN, Warszawa 1974.

[27] Fridley, J.C. Analiza dynamiki procesów. WNT, Warszawa, 1974.

[28] Gabasov, P., Kirillova, F.N. Osnovy dinamičeskogo programmirovanija. Izdatel'stvo BGU. Minsk, 1975.

[29] Garfinkel, E.S., Nemhauser G.L. Programowanie całkowitoliczbowe. PWN, Warszawa 1978.

[30] Glover, F., Klingman, D.On the equivalence of some generalized problems to pure network problems. Mathemalical Programming, Vol.4, 1973.

[31] Goetz, J.Zagadnienie przydziału zasobów i szeregowania operacji w systemie cyfrowym /rozprawa doktorska/. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Raport 774. Wrocław, 1978.

[32] Gościński, A. Komputerowe systemy sterowania złożonymi dyskretnymi procesami produkcyjnymi. Zeszyty Naukowe AGH. Automatyka. Zeszyt 14, 1976.

[33] Gościński, A. Zieliński, K. Optymalne szeregowanie programów użytkowych w maszynach cyfrowych pracujących w czasie rzeczywistym. Archiwum Automatyki i Telemechaniki, Tom XXII, Z.3, 1977.

[34] Gościński, A., Zieliński, K. Wpływ kosztów sterowania na pogorszenie jakości procesu i częstotliwości aktywizacji programów użytkowych. Archiwum Automatyki i Telemechaniki, Tom XXIII, Z.1-2, 1978.

[35.] Grabowski, J. Uogólnione zagadnienia optymalizacji kolejności operacji w dyskretnych systemach produkcyjnych. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, Wrocław, 1979.

[36.] Grudzewski, W. Szamkołowicz, L. Zastosowanie teorii grafów i metod sieciowych w planowaniu przedsięwzięć organizacyjno - technicznych.
Politechnika Wrocławska, Wrocław, 1974.

[37.] Ingargiola, G.P., Korsh, J.K. Reduction algorithm for 0-1 single knapsack problem. Man.Sci, Vol.20, No.4, 1973.

[38.] Janiak, A., Kaczmarek, W., Stanisz, A. System operacyjny dla zestawu ODRA 1325 SMA do sterowania eksperymentem badawczym w czasie rzeczywistym. Część II. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Raport 166. Wrocław, 1976.

[39.] Jewell, W.S. Optimal flows through networks with gains. Operations Research, Vol.10, 1962.

[40.] Katzan, H, jr. Operating systems. A pragmatic approach. Van Nostrand, 1973.

[41.] Kaufmann, A., Henry - Labordere, A. Methodes et modeles de la recherche operationelle, Dunod, Paris, 1973.

[42.] Kianfor, F. Stronger inequalities for 0-1 integer programming Camputational references. Operations Research. Vol.24,No 3. 1978.

[43.] Kordecki,H., Koszałka,L., Kurzyński,M.,Myszka,W., Popkiewicz,M. Komputerowy system automatyzacji eksperymentu. Prace VIII KKA, Szczecin /w druku/.

[44.] Kordecki,H. Zarządzanie pamięcią operacyjną komputerowych systemów sterowania w przypadku deterministycznym. Prace VIII KKA, Szczecin /w druku/.

[45.] Korzan, B. Elementy teorii grafów i sieci. Metody i zastosowania. WNT, Warszawa, 1978.

[46.] Kovalev, M.M. Diskretnaja optimizacija. Izdatel'stvo BGU, Mins, 1977.

[47.] Lipaev, V.V. Raspredelenie resursov v vycislitel'nych sistemach. Statistika, Moskva, 1979.

[48.] Lipaev, V.V., Kolin, K.K. Oprogramowanie podstawowe maszyn cyfrowych. PWN, Warszawa, 1977.

[49.] Liu, G.L., Layland, J.W. Scheduling algorithims for multiprogramming in a hard - real - time environment - J.ACM, Vol.20, No.1, 1973.

[50.] Madnick, S.E., Donovan, J.J. Operating systems. Mc Graw Hill, 1974.

[51.] Martin, J. Programming real - time systems. Prentice - Hall, 1965.

[52.] Martin, J. Design of real - time computer systems. Prentice - Hall, 1967.

[53.] Mielcarek, J. Adaptacyjna obsługa zgłoszeń w systemach operacyjnych czasu rzeczywistego /rozprawa doktorska/. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Komunikat 554, 1977.

[54.] Moiseev, N.N, Elementy teorii optimal'nych sistem. Izdatel'stvo Nauka, Moskva.

[55.] Myszka, W., Rozdział zasobów systemu cyfrowego optymalny ze względu na jakość sterowania procesem technologicznym. Materiały VIII KKA, Szczecin /w druku/.

[56.] Myszka, W., Algorytmy rozdziału zasobów /w przygotowaniu/.

[57.] Nauka Lemke, von,H.R., Verbruggen, H.B. /red/. Digital computer applications to process control. Proceedings of the 5th IFAC - IFIP International Conference, The Hague The Netherlands, June 14-17, 1977.

[58.] Niederliński, A. Systdmy cyfrowe automatyki przemysłowej. Tom 1. Sprzęt i oprogramowanie, WNT, Warszawa 1977.

[59.] Niederliński, A. Systemy cyfrowe automatyki przemysłowej. Tom 2. Zastosowania. WNT, Warszawa, 1977.

[60.] Opis procesu technologicznego produkcji kwasu siarkowego. Ministerstwo Przemysłu Chemicznego. Zakład Chemiczny POLICE, Police, 1974 /mat.pow./

[61.] Poradnik inżyniera i technika. Kwas siarkowy. WNT, 1975.

[62.] Rafajłowicz,E. Identyfikacja pewnej klasy obiektów o parametrach rozłożonych metodą największej wiarogodności. (Rozprawa doktorska).Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Raport Pre 70, Wrocław, 1979.

[63.] Respondek, J. Podstawy inżynierii reakcji chemicznych. Politechnika Wrocławska, Wrocław, 1977.

[64.] Saffer, S.I., Mishelevich. D.J. A definition of real time computing. CACM, Vol.18, No 9, 1975.

[65.] Show, A.C. The logical design of operating systems. Prentice Rall, 1974.

[66.] Truemper, K. An efficient scaling procedure for gain networks. Networks, Vol 6, 1976.

[67.] Truemper, K. On max flaws with gains and pure min cost flows. SIAM J.Appl. Math. Vol. 32, No.2, 1977.

[68.] Truemper, K. Optimal flows in nonlinear gain networks. Networks, Vol.8, No.1, 1978.

[69.] Tsichritzis, D.C. Bernstein, P.A. Operating systems. Academic Press, 1974.

[70.] Węglarz, J. Błażewicz, J., Cellary, W., Słowiński, R. Algorithm 520.An automatic revised simplex method for constrained network scheduling.
ACM Trans - on Math. Sottware, Vol.3,N.3, 1977.

[71] Węgrzyn, S. Zarys podstaw kierowania. Systemy operacyjne czasu rzeczywistego. Podstawy Sterowania, T.3, Z.4, 1973.

[72] Yourdon, E. Projektowanie systemów o działaniu bezpośrednim, WNT, Warszawa 1976.

Niniejszy raport otrzymują:

1.	OINT	1 egz.
2.	Biblioteka międzyinstytutowa i OINT I-6; I-25	1 egz.
3.	Promotor	1 egz.
4.	Recenzenci	2 egz.
5.	Instytut Automatyki, Politechnika Warszawska	1 egz.
6.	Archiwum I-6	1 egz.
7.	Autor	3 egz.
	Razem:	10 egz.