Politechnika Wrocławska
Wydział Informatyki i Zarządzania

ROZPRAWA DOKTORSKA

Zastosowanie metod programowania z ograniczeniami w zadaniowo zorientowanych systemach interakcyjnego wspomagania decyzji

mgr inż. Grzegorz Bocewicz

Promotor:
Prof. dr hab. inż. Zbigniew Banaszak

Wrocław 2007

Serdecznie dziękuję mojemu promotorowi prof. dr hab. inż. Zbigniewowi Banaszakowi za cenne rady i sugestie oraz wszelką pomoc podczas realizacji niniejszej pracy.

Spis treści

Spis ilustracji

Spis przykładów

Spis ważniejszych oznaczeń

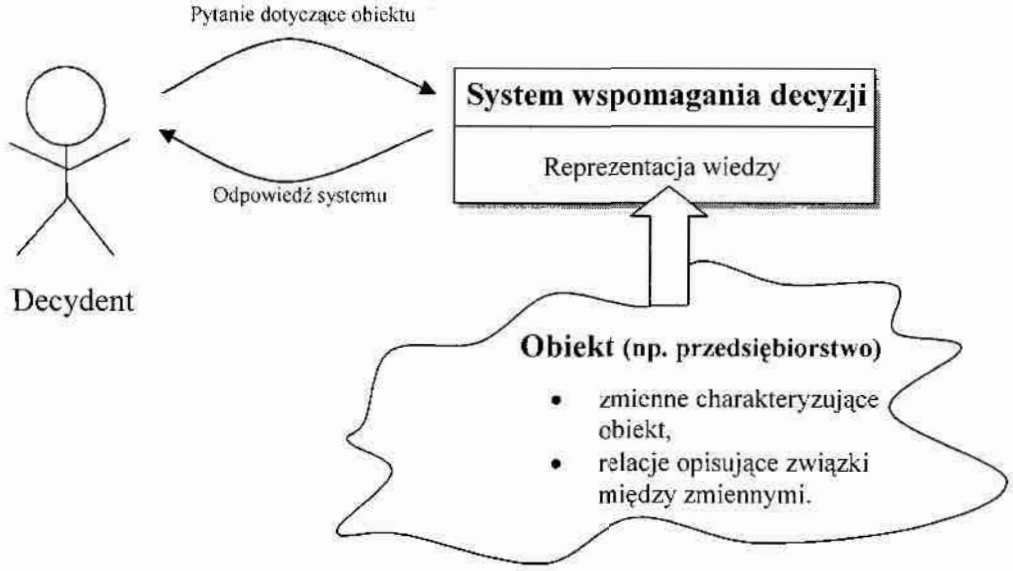
Symbol	Znaczenie
α	ciąg formuł elementarnych: $(\alpha_1,\alpha_2, \ldots ,\alpha_N)$
N	liczba formuł elementarnych
αu	ciąg wejściowych formuł elementarnych: $(\alpha u_1, \alpha u_2, \ldots , \alpha u_k)$
αw	ciąg pomocniczych formuł elementarnych: $(\alpha w_1, \alpha w_2, \ldots , \alpha w_r)$
αy	ciąg wyjściowych formuł elementarnych: $(\alpha y_1, \alpha y_2, \ldots , \alpha y_p)$
a	ciąg wartości logicznych formuł elementarnych: $(a_1, a_2, \ldots , a_N)$
au	ciąg wartości logicznych wejściowych formuł elementarnych: $(au_1, au_2, \ldots ,au_k)$
aw	ciąg wartości logicznych pomocniczych formuł elementarnych: $(aw_1, aw_2, \ldots ,aw_r)$
ay	ciąg wartości logicznych wyjściowych formuł elementarnych: $(ay_1, ay_2, \ldots ,ay_p)$
u	ciąg zmiennych wejściowych: $(u_1, u_2, \ldots , u_{ku})$
w	ciąg zmiennych pomocniczych: $(w_1, w_2, \ldots , w_{kw})$
y	ciąg zmiennych wyjściowych: $(y_1, y_2, \ldots , y_{ky})$
pr	ciąg parametrów: $(pr_1, pr_2, \ldots , pr_{kr})$
KB	reprezentacja wiedzy
$F(\alpha)$	zbiór faktów opisujących obiekt: $\{F_1(\alpha), F_2(\alpha), \ldots , F_K(\alpha)\}$
$Q(a)$	ciąg wartości logicznych faktów $F(\alpha)$
K	liczba faktów
$Fu(\alpha u)$	zbiór faktów wejściowych: $\{Fu_1(\alpha u), Fu_2(\alpha u), \ldots , Fu_{Pf}(\alpha u)\}$
$Qu(au)$	ciąg wartości logicznych faktów $Fu(\alpha u)$
Pf	liczba faktów wejściowych
$Fy(\alpha y)$	zbiór faktów wyjściowych: $\{Fy_1(\alpha y), Fy_2(\alpha y), \ldots , Fy_{Rf}(\alpha y)\}$
$Qy(ay)$	ciąg wartości logicznych faktów $Fy(\alpha y)$
Rf	liczba faktów wyjściowych
$Fu(u)$	zbiór wartości logicznych faktów wejściowych w zależności od zmiennych u (właściwość wejściowa)
$Qu(u)$	ciąg wartości logicznych faktów $Fu(u)$
$Fy(y)$	zbiór wartości logicznych faktów wyjściowych w zależności od zmiennych y (właściwość wyjściowa)
$Qy(y)$	ciąg wartości logicznych faktów $Fy(y)$
Re	relacja będąca zbiorem wszystkich trójek (u,w,y), dla których wszystkie fakty KB są prawdziwe
$F(u,w,y)$	zbiór wartości logicznych faktów w zależności od zmiennych u, w, y
$Q(u,w,y)$	ciąg wartości logicznych faktów $F(u,w,y)$
$F_P(u,w,y, pr)$	schemat faktów - zbiór wartości logicznych faktów w zależności od zmiennych u,w,y i parametrów pr
$F_a(\Theta, S_0, x, P_D)$	zbiór faktów opisujących podstawowe założenia systemów transportowych
$Q_a(\Theta, S_0, x, P_D)$	ciąg wartości logicznych faktów $F_a(\Theta, S_0, x, P_D)$
$F_b(\Theta, S_0, x, P_D)$	zbiór faktów dodatkowych, opisujących indywidualne cechy rozważanego systemu
$Q_b(\Theta, S_0, x, P_D)$	ciąg wartości logicznych faktów $F_b(\Theta, S_0, x, P_D)$
$F_{Pa}(\Theta, S_0, x, P_D, pr)$	schemat faktów reprezentujący podstawowe założenia systemów transportowych
PSO	problem spełniania ograniczeń: $((V,D), C)$
V	zbiór dyskretnych zmiennych decyzyjnych: $\{V_1, V_2, \ldots ,V_n\}$
V_i	i-ta dyskretna zmienna decyzyjna
C	zbiór ograniczeń: $\{C_1, C_2, \ldots ,C_{Lc}\}$
C_i	i-te ograniczenie
C_P	schemat ograniczeń, zbiór ograniczeń sparametryzowanych: $\{C_1(pr), C_2(pr), \ldots , C_r(pr)\}$
$C_i(pr)$	i-te ograniczenie sparametryzowane, zależne od wartości parametrów pr
D	zbiór dziedzin zmiennych decyzyjnych: $\{D_1, D_2, \ldots ,D_n\}$
D_i	dziedzina i-tej zmiennej decyzyjnej
$RPSO$	rozmyty problem spełniania ograniczeń: $((V_r, D_r), C_r)$
V_r	zbiór rozmytych zmiennych decyzyjnych: $\{V_{r,1}, V_{r,2},\ldots,V_{r,n}\}$
$V_{r,i}$	i-ta rozmyta zmienna decyzyjna

Symbol	Znaczenie
$\mu_i(v)$	funkcja przynależności i-tej decyzyjnej zmiennej rozmytej $V_{r,i}$
D_r	zbiór rozmytych zmiennych decyzyjnych: $\{D_{r,1}, D_{r,2}, \ldots, D_{r,n}\}$
$D_{r,i}$	dziedzina i-tej rozmytej zmiennej decyzyjnej
PSO_{Ii}	problem spełniania ograniczeń reprezentujący zgodnie z interpretacją I_i rozmyty problem spełniania ograniczeń $RPSO$
I	zbiór interpretacji zmiennych rozmytych: $\{I_1, I_2, \ldots, I_{qn}\}$
I_i	i-ta interpretacja zmiennych rozmytych
$\mu_{i,j}(v)$	funkcja przynależności j-tej decyzyjnej zmiennej rozmytej $V_{r,j}$, w i-tej interpretacji
$\{a_{f,1}, \ldots, a_{f,rf}\}_{Ii,j}$	zbiór parametrów charakteryzujących funkcję przynależności $\mu_{i,j}(v)$
$a_{f,i}$	i-ty parametr charakteryzujący funkcję przynależności
a^{Ii}	zbiór dyskretnych zmiennych decyzyjnych dla PSO_{Ii} (zbiór parametrów $a_{f,i}$ odpowiadających interpretacji I_i)
D^{Ii}	zbiór dziedzin zmiennych dla PSO_{Ii}
C^{Ii}	zbiór ograniczeń dla PSO_{Ii}
P	zbiór procesów: $\{P_1, P_2, \ldots, P_q\}$
P_i	i-ty proces (i-ta marszruta): $(R_o, R_p, \ldots, R_r)$
m_i	liczba zasobów wchodzących w skład P_i
q	liczba procesów realizowanych w systemie
p_{ij}	j-ta operacja elementarna procesu P_i
po	sekwencja operacji realizowanych w systemie
S	sekwencja stanów: $(S_0, S_1, \ldots, S_w)$
S_0	stan początkowy: $(R^{s0}_1, R^{s0}_2, \ldots, R^{s0}_q)$
S_i	i-ty stan systemu,
$S0$	zbiór stanów początkowych S_0
R	zbiór zasobów: $\{R_1, R_2, \ldots, R_{ks}\}$
R_i	i-ty zasób
N_i	liczba procesów realizowanych na współdzielonym zasobie R_i
$R_{L,i}$	i-ty zasób lokalny: $R_{L,i} \in R$
$R_{w,i}$	i-ty zasób współdzielony: $R_{w,i} \in R$
R^{s0}_i	zasób, od którego rozpoczyna pracę proces P_i : $R_{w,i} \in R$
R^{si}_j	zasób na którym proces P_j realizuje operacje w stanie S_i
Θ	sekwencja reguł priorytetowania: $(\sigma_1, \sigma_2, \ldots, \sigma_l)$
O	zbiór sekwencji reguł priorytetowania Θ
σ_i	i-ta reguła priorytetowania, sekwencja procesów obsługiwanych na zasobie R_i: $(P_1, P_3, \ldots, P_r)$
Sp_i	sekwencja, procesów obsługiwanych na zasobie R_i przed procesami sekwencji σ_i
x	sekwencja terminów rozpoczęcia realizowanych operacji (harmonogram pracy)
X	zbiór harmonogramów x
x_i	termin rozpoczęcia operacji i-tej operacji sekwencji po
x^j_s	sekwencja terminów $x_{i,j}$ rozpoczęcia operacji realizowanych przez proces P_i
x^j_σ	sekwencja terminów rozpoczęcia operacji realizowanych przez procesy realizowane na zasobie R_j
t	sekwencja czasu trwania operacji elementarnych
t_i	czas trwania operacji odpowiadającej operacji o terminie rozpoczęcia x_i
P_D	sekwencja zmiennych pomocniczych
PD	zbiór wartości zmiennych pomocniczych
W	okno czasowe
No_i	sekwencja terminów rozpoczęcia elementarnych operacji, procesów obsługiwanych przez zasób R_i
Ws_i	sekwencja procesów obsługiwanych przez zasób współdzielonych R_i
KB_{kom}	skompresowana reprezentacja wiedzy
$F_{a\,kom}(\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom})$	skompresowana postać faktów F_a
$Q_{a\,kom}(\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom})$	ciąg wartości logicznych faktów $F_{a\,kom}(\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom})$
Θ_{kom}	skompresowana sekwencja reguł priorytetowania

Symbol	Znaczenie
O_{kom}	zbiór sekwencji reguł priorytetowania Θ_{kom}
$\sigma_{ZW,i}$	i-ta zastępcza reguła priorytetowania
$S_{0\ kom}$	skompresowana sekwencja stanów początkowych
$S0_{kom}$	zbiór stanów początkowych $S_{0\ kom}$
x_{kom}	skompresowana sekwencja x
X_{kom}	zbiór harmonogramów x_{kom}
$P_{D\ kom}$	skompresowana sekwencja zmiennych pomocniczych
PD_{kom}	zbiór sekwencji $P_{D\ kom}$
$R_{ZL,i}$	i-ty zastępczy zasób lokalny
$R_{ZW,i}$	i-ty zastępczy zasób współdzielony
R_g	gałąź struktury SWPC, zbiór zasobów tworzących ścieżkę, na których realizowane są te same procesy
Gp	zbiór postaci grafów żądań zasobowych
G	sekwencja grafów żądań zasobowych: $(G^{s0}, G^{s1}, G^{s2}, \ldots, G^{sw})$
$G^{si} = (N^{si}, B^{si})$	graf żądań zasobowych
N^{si}	zbiór wierzchołków grafu G^{si}
$R^{si}_{n,j}$	j-ty wierzchołek grafu G^{si}
$\psi(R^{si}_{n,j})$	współrzędna (etykieta) wierzchołka $R^{si}_{n,j}$
B^{si}	zbiór łuków grafu G^{si}
$B^{si}_{k,l}$	łuk grafu G^{si} łączący wierzchołek R_k z wierzchołkiem R_l
$\varphi(B^{si}_{k,l})$	współrzędna (etykieta) łuku $B^{si}_{k,l}$
R_k	wierzchołek (zasób) będący początkiem łuku $B^{si}_{k,l}$
R_l	wierzchołek (zasób) będący końcem łuku $B^{si}_{k,l}$
$\Delta(G^{si})$	operator spełnienia żądań zasobowych
${}^{p}x_v^{s(i-1)}(x)$	współrzędna łuku, grafu $G^{s(i-1)}$, reprezentującego p-ty proces w stanie S_{i-1}
${}^{p}x_v^{si}$	termin rozpoczęcia operacji p-tego procesu na zasobie R_k w stanie S_i, współrzędna luku reprezentującego p-ty proces w grafie G^{si}
${}^{p}t_v^{si}$	czas trwania operacji reprezentowanej przez ${}^{p}x_v^{si}$
$R_{p,si}$	zasób, na którym realizowana jest operacja o terminie rozpoczęcia ${}^{p}x_v^{si}$
P_p	proces reprezentowany przez termin rozpoczęcia ${}^{p}x_v^{si}$
P_e	proces występujący na zasobie $R_{p,si}$ przed procesem P_p
${}^{pe}x_v^{sj}$	współrzędna łuku opisującego proces P_e w grafie reprezentującym stan S_f
${}^{z}G^{si}$	z-ty spójny podgraf grafu G^{si}
${}^{z}N$	zbiór wierzchołków grafu ${}^{z}G^{si}$
${}^{z}B$	zbiór łuków grafu ${}^{z}G^{si}$
${}^{z}x^{si}$	zbiór współrzędnych wszystkich łuków wchodzących w skład podgrafu ${}^{z}G^{si}$
L_{Ni}	liczba węzłów na i-tym poziomie dystrybucji
L_P	liczba poziomów dystrybucji (liczba poziomów drzewa potencjalnych rozwiązań)
$R^{si}_{n,j}$	j-ty wierzchołek grafu G^{si} – zasób, na którym w stanie S_i realizowane są operacje lub zasób do którego w stanie S_i procesy żądają dostępu
$\varepsilon(C)$	funkcja określająca stopień zawężenia drzewa potencjalnych rozwiązań wyniku zadanego zbioru ograniczeń C
R_T	rozmiar drzewa potencjalnych rozwiązań dla czterech poziomów dystrybucji
R_{T1}	rozmiar drzewa potencjalnych rozwiązań dla dowolnej liczby poziomów dystrybucji
R_{T2}	rozmiar drzewa potencjalnych rozwiązań, dla dowolnej liczby poziomów dystrybucji, przy zastosowaniu dwuetapowej strategii poszukiwania ma postać
Z	liczba kroków obliczeniowych koniecznych do rozwiązania problemu PSO, któremu odpowiada drzewo potencjalnych rozwiązań o rozmiarze R_T
Z_1	liczba kroków konieczna do wyznaczania wszystkich rozwiązań dopuszczalnych problemu PSO_{Su1} (lub PSO_{Su1})
Z_2	liczba kroków konieczna do wyznaczania wszystkich rozwiązań dopuszczalnych problemu PSO_{Su1} (lub PSO_{Su1}) przy zastosowaniu dwuetapowej strategii przeszukiwania
Z_x	liczba kroków obliczeniowych Z_x konieczna do wyznaczenia elementów sekwencji x
ujc	umowna jednostka czasu
ujd	umowna jednostka długości
G_T	współczynnik nasycenia systemu SWPC
$G_{T,i}$	współczynnik nasycenia i-tego procesu realizowanego w systemie SWPC

1. Wstęp

Zagadnienia poruszane w pracy dotyczą problematyki projektowania zadaniowo zorientowanych, interakcyjnych systemów wspomagania decyzji. Cechą charakterystyczną tego rodzaju systemów jest możliwość wspomagania decydenta w zadanym obszarze jego działalności.



Rys. 1.1. Idea wspomagania decyzji

Na rysunku 1.1 przedstawiona została ogólna idea wspomagania decyzji. **Obiekt** (na przykład przedsiębiorstwo produkcyjne, system transportowy, system składowania itp.) – jego struktura i zachowanie są charakteryzowane zbiorami zmiennych decyzyjnych i ich dziedzin oraz łączących je zbiorów relacji (ograniczeń). Obiekt w systemie wspomagania decyzji specyfikowany jest w postaci reprezentacji wiedzy. Otoczenie (**decydent**) komunikuje się z systemem formułując **zdania** (pytania) w postaci relacji łączących wybrany zbiór zmiennych decyzyjnych (struktura pytania będzie szerzej omówiona dalszych rozdziałach). **System wspomagania decyzji** komunikuje się z użytkownikiem formułując **zdania** (odpowiedzi) zawierające zbiory wartości zmiennych spełniających ograniczenia (relacje) zadane przez użytkownika oraz ograniczenia charakteryzujące obiekt. Obecnie spotykane systemy umożliwiają udzielanie odpowiedzi na z góry zadane zbiory pytań. Wspomaganie odbywa się dla obiektów należących do wspólnej klasy. Na przykład popularny pakiet **OptiTrans** umożliwia wspomaganie decyzji z zakresu planowania transportu i jest w stanie udzielać odpowiedzi na pytania związane bezpośrednio z planowaniem tras, rozładunku środków transportu i innych cech charakterystycznych dla klasy problemów transportowych.

Rozwój technologii informatycznych spowodował, że komputerowe systemy wspomagania spełniają współcześnie istotną rolę w procesach decyzyjnych, szczególnie tam, gdzie konieczne jest szybkie podejmowanie decyzji (np. w przedsiębiorstwach). Systemy wspomagania decyzji odgrywają szczególną rolę w procesach planowania zleceń

produkcyjnych, składowania, marszrutowania, itp. Procesy te stanowią zwykle integralną całość określonego procesu produkcyjnego [29], [81], [53]. Ze względu na ich specyfikę warunkowaną szybko zachodzącymi zmianami pracy stosowanych obiektów obserwowana jest tendencja projektowania systemów o charakterze interakcyjnym. Przez interakcyjność rozumiana jest taka zdolność systemu wspomagania decyzji, do przetwarzania danych wprowadzanych (zadanych) przez decydenta oraz udzielania odpowiedzi na zadawane pytania w akceptowalnym przez niego czasie (zwykle w trybie na bieżąco).

Obecnie systemy komercyjne udostępniają szeroki zakres narzędzi modelowania spotykanych w praktyce problemów (takich jak problemy magazynowania, kalkulacji kosztów, weryfikacji zleceń produkcyjnych), okazuje się jednak, że w wielu przypadkach narzędzia te są niewystarczające [29], [30], [31], [24]. Warto zauważyć, że rozwiązania komercyjne zwykle „bardzo trudno" poddają się modyfikacjom (przeprogramowaniu modułu optymalizacji, dostosowaniu systemu do indywidualnego wariantu problemu, itp. [24]). Wynika to głównie ze skomplikowanej budowy modułów obliczeniowych, które wykorzystują szereg wzajemnie współdziałających heurystyk.

Na uwagę zasługuje również fakt, iż większość dostępnych systemów nastawionych jest na poszukiwanie **rozwiązań optymalnych**. Przy rozwiązywaniu problemów optymalnych zakłada się (zwykle niejawnie) istnienie niepustego zbioru **rozwiązań dopuszczalnych** [24] (spełniających jednocześnie wszystkie ograniczenia opisujące obiekt i ograniczenia zadane przez decydenta). Okazuje się, że w wielu spotykanych w praktyce problemach (np. harmonogramowaniu pracy wózków samojezdnych w systemach transportowych dopuszczających występowanie blokad), rozstrzyganie o istnieniu rozwiązania dopuszczalnego jest tak samo trudne, jak ocena odległości uzyskanego rozwiązania od rozwiązana optymalnego. Zatem wyznaczenie rozwiązania optymalnego w wielu praktycznych zastosowaniach okazuje się niemożliwe w żądanym przez decydenta czasie.

Potrzeba poszukiwania rozwiązań dopuszczalnych jest szczególnie widoczna w problemach timetabling'u, czy szerzej w problemach planowania przedsięwzięć w warunkach występowania czasowych ograniczeń dostępu do zasobów. Skalę tych problemów ilustruje konieczność przeszukiwania przestrzeni wzajemnie przenikających się problemów rozmieszczania, marszrutowania, porcjowania i harmonogramowania, wzajemnie warunkujących się poziomów procesów technologicznych i transportowo-magazynowych, itp.

W kontekście tej klasy problemów perspektywicznym aparatem, który z powodzeniem może być stosowany w systemach wspomagania decyzji do modelowania obiektu i poszukiwania rozwiązań, są techniki programowania z ograniczeniami (ang. Constraint Programming, *CP*) implementowane w pakietach takich jak np. **Chip** [33], **Ilog** [99] czy **Oz Mozart** [77]. Technik te wykorzystywane są do rozwiązywania problemów spełniania ograniczeń (*PSO*). Potwierdziły wielokrotnie swą użyteczność w zakresie spotykanych w praktyce problemów decyzyjnych [81], [29], [53], [24], [47]. Deklaratywny charakter tych języków, pozwala w większym stopniu koncentrować się na ograniczeniach specyfikujących problemy, niż na samych metodach ich rozwiązywania. Droga od sformułowania problemu do jego rozwiązania składa się z dwóch etapów:

- sformułowania rozważanego problemu jako problemu spełniania ograniczeń (*PSO*),
- implementacji *PSO* za pomocą środków dostarczanych przez język określonego systemu programowania z ograniczeniami.

Wykorzystanie technik programowania z ograniczeniami do wspomagania decyzji (wyznaczenia rozwiązań dopuszczalnych) wymaga posiadania specyfikacji problemu w postaci zbioru ograniczeń. Okazuje się, że wykorzystanie reprezentacji wiedzy, do opisu, budowy oraz funkcjonowania obiektów, pozwala oceniać w ramach jednego mechanizmu wnioskowania, różne specyfikacje problemu. Pozwala to na poszukiwanie takiej specyfikacji, która zawierałaby warunki wystarczające spełnienie których gwarantuje istnienie odpowiedzi na zadany zbiór pytań rutynowych.

W takim ujęciu obiekt (przedsiębiorstwo, system produkcyjny, podsystem transportowy, itp.) może być postrzegany jako wiedza (zbiór zmiennych decyzyjny, relacji i opisów), służąca za platformę, dla formułowania pytań, jak też i wypracowywania odpowiedzi. Istotne w tym kontekście wydaje się pytanie: Czy system wspomagania decyzji wykorzystujący daną bazę wiedzy jest w stanie udzielić odpowiedzi na zbiór potencjalnych pytań rutynowych? (inaczej mówiąc: Czy w zadanej bazie wiedzy istnieją warunki gwarantujące istnienie odpowiedzi na zbiór potencjalnych pytań ?).

Nietrudno zauważyć, że zarówno pytania jak i poszukiwane odpowiedzi muszą być sformułowane w terminach (parametrów i/lub zmiennych) występujących w dostępnej bazie wiedzy. Przyjmując to założenie, poszukiwane są odpowiedzi na pytania typu: Czy każde pytanie rutynowe ma swoją odpowiedź? Jeżeli nie to, jakie uzupełnienie (rozszerzenie) bazy wiedzy ją gwarantuje? Jakie alternatywne podzbiory zmiennych decyzyjnych, dla jakich kombinacji wartości swoich zmiennych, gwarantują osiągnięcie zadanych wartości wybranego podzbioru zmiennych decyzyjnych?

Pytania te wiążą się z kwestiami weryfikacji bazy wiedzy. Na przykład, odpowiedź na pierwsze z wyżej wymienionych pytań wiąże się z weryfikacją jej spójności (gdzie spójność bazy wiedzy oznacza istnienie warunków spełnienie, których gwarantuje spełnienie określonych właściwości). W ogólnym przypadku kwestie te dotyczą badania spójności, niesprzeczności i nadmiarowości bazy wiedzy. Podejście takie pozwala więc na budowanie i weryfikowanie baz wiedzy (specyfikacji problemów w postaci ograniczeń) pod kątem istnienia warunków wystarczających, spełnienie których gwarantuje istnienie odpowiedzi na zadany zbiór pytań (istnienie rozwiązania dopuszczalnego).

Perspektywicznym narzędziem do opisu funkcjonowania obiektu w postaci reprezentacji wiedzy, a tym samym do specyfikacji rozwiązywanego problemu jest metoda logiczno-algebraiczna [25], [27], [28]. Formalizm tej metody oraz dostępne metody wnioskowania pozwalają na poszukiwanie warunków gwarantujących istnienie odpowiedzi na zadane pytania rutynowe.

Dostępność specyfikacji problemu zawierającej warunki wystarczające umożliwia, budowę efektywnych (gwarantujących istnienie rozwiązań dopuszczalnych) systemów interakcyjnego wspomagania decyzji. W przedstawionym kontekście, niniejsza praca przedstawia propozycję wykorzystania technik programowania z ograniczeniami oraz metody

logiczno–algebraicznej jako efektywnych narzędzi do budowy interakcyjnych, zadaniowo zorientowanych systemów wspomagania decyzji.

1.2. Cel i zakres pracy

Systemy wspomagania decyzji przeznaczone do rozwiązywania takich problemów jak problemy marszrutowania, harmonogramowania, porcjowania, itp. wymagają specyfikacji problemów w postaci gwarantującej istnienie odpowiedzi na zadany zbiór pytań rutynowych. Jednocześnie wymagają poszukiwania efektywnych strategii przeszukiwania, umożliwiających uzyskanie odpowiedzi w trybie interakcyjnym. Ograniczeniami w spełnianiu przyjętych wymagań są tutaj zasady funkcjonowania i cechy obiektów, dla których prowadzone jest wnioskowanie, wyrażane w postaci opisowej. Są to takie ograniczenia jak: zasady poruszania się wózków w określonej strukturze sytemu transportowego, zasady magazynowania określonego rodzaju wyrobów, zasady układania kontenerów w magazynie, itp. Oczywiście, analogiczne ograniczenia mogą być formułowane dla innych problemów niż planowanie produkcji, jak na przykład w problemach sterowania ruchem w określonej sieci komputerowej, itp. Wspólnym celem jest jednak określenie sposobu szybkiego prototypowania rozwiązań, gwarantujących spełnienie określonych żądań użytkownika systemu.

Stosowane metody, w szczególności metody optymalizacji i/lub symulacji, charakteryzują się dużą czasochłonnością, pracochłonnością oraz wysokimi kosztami zastosowania. Obsługiwanie systemów opartych na tych metodach wymaga zwykle od użytkownika wysokich kwalifikacji. Wdrażanie tych metod oddala zatem możliwość wyznaczenia rozwiązań dopuszczalnych w trybie interakcyjnym.

Metody programowania z ograniczeniami, pomimo rosnącej liczby zastosowań, są wciąż mało znane [53]. Bazują one na procedurach propagacji ograniczeń i dystrybucji zmiennych (stanowiących podstawowe elementy strategii poszukiwania rozwiązań), których naprzemienne stosowanie pozwala w szybki sposób wyszukać rozwiązanie dopuszczalne (zbiór rozwiązań alternatywnych). Metody te stanowią interesującą alternatywę wobec aktualnie wykorzystywanych metod (programowania matematycznego, metod heurystycznych, algorytmów ewolucyjnych, itp.), w rozwiązywaniu problemów o charakterze decyzyjnym.

Cel:

Celem pracy jest opracowanie metody projektowania zadaniowo zorientowanych systemów interakcyjnego wspomagania decyzji.

Opracowana metoda dotyczy projektowania systemów wspomagania decyzji, przeznaczonych dla obiektów opisywanych przez zmienne dyskretne, w których wspomaganie decyzji polega na rozwiązywaniu problemów kombinatorycznych (planowanie, harmonogramowanie, marszrutowanie, składowanie itp.) spotykanych powszechnie w przedsiębiorstwach produkcyjnych.

Zakres pracy obejmuje analizę wybranego problemu wspomagania decyzji w podsystemach transportowych elastycznych systemów produkcyjnych (ESP) [95], tzn., wspomagania procesu harmonogramowania pracy wózków samojezdnych. W ramach pracy opracowano przykłady ilustrujące możliwości i zastosowanie metody logiczno-algebraicznej implementowanej w technikach *CP*, w systemach wspomagania decyzji. Opracowano model interakcyjnego systemu wspomagania decyzji. Przedstawiono procedurę budowy sparametryzowanych struktur zbiorów ograniczeń zwanych schematami ograniczeń (faktów). Na wybranych przykładach problemów magazynowania zilustrowano wykorzystanie schematów ograniczeń do budowy specyfikacji różnego rodzaju problemów spełniania ograniczeń (ostrych i rozmytych). W oparciu o metodę logiczno-algebraiczną i techniki programowania z ograniczeniami opracowano procedurę wyznaczania warunków wystarczających gwarantujących istnienie odpowiedzi na zadany zbiór rutynowych pytań.

Ponadto, zaproponowano efektywną strategię przeszukiwania przestrzeni potencjalnych rozwiązań. W ramach tej strategii przeprowadzono badania porównawcze z istniejącymi rozwiązaniami (mechanizmy wnioskowania metody logiczno-algebraicznej). Dla zadanej klasy systemów transportowych pokazano twierdzenie gwarantujące istnienie harmonogramów bezblokadowych. W oparciu o to twierdzenie, dla rozważanej klasy systemów transportowych, opracowano model schematu faktów. Ponadto opracowano metody kompresji schematów faktów pozwalające na zmniejszenie niezbędnej liczby faktów i zmiennych wchodzących w skład bazy wiedzy.

W kontekście rozwiązywania problemów harmonogramowania, przeprowadzono badania skuteczności zaproponowanych rozwiązań.

Praktycznym wynikiem pracy jest komputerowy system sterowania dyspozytorskiego. Pakiet ten opracowano przy użyciu języka programowania **Java**, z wykorzystaniem mechanizmów języka programowania ograniczeń **Oz Mozart** (stanowiącego moduł obliczeniowy). System umożliwia rozwiązywanie typowych problemów decyzyjnych występujących w określonej klasie systemów transportowych, w trybie interakcyjnym. Pozwala między innymi, na wyznaczanie bezblokadowych harmonogramów pracy wózków samojezdnych, spełniających zadane przez użytkownika ograniczenia.

1.2.1. Teza

Na etapie projektowania systemów wspomagania decyzji, istotna jest ocena możliwości implementacji i wykorzystania w tych systemach technik *CP*, przy szczególnym uwzględnieniu wykorzystania metody logiczno-algebraicznej jako narzędzia do reprezentacji wiedzy. Wymagane jest by rozważane systemy gwarantowały istnienie odpowiedzi na zadane przez użytkownika pytania. Dodatkowo, systemy te powinny zapewniać odpowiedni stopień szczegółowości specyfikacji obiektu uwzględniającej zadane przez użytkownika żądania w kontekście wypracowywanych decyzji.

Zważywszy na powyższe oraz uwzględniając fakt dostępności pakietów implementujących techniki *CP* typu *public domain* (np. **Oz Mozart**), teza pracy została sformułowana następująco:

Teza:

Implementacja metody logiczno-algebraicznej w technikach programowania z ograniczeniami umożliwia budowę dedykowanych systemów interakcyjnego wspomagania decyzji.

1.3. Sformułowanie problemu

Dany jest obiekt (np. podsystem transportowy elastycznego systemu produkcyjnego), którego budowa i działanie opisane jest w postaci bazy wiedzy (zwierającej wiedzę dotyczącą na przykład: struktury systemu, zbioru zasad panujących w systemie takich jak: reguły poruszania się wózków, kolejność obsługi wózków na zasobach współdzielonych, itd.). Znany jest zbiór pytań rutynowych (opcji, np. Jaki harmonogram gwarantuje realizację założonych operacji w zadanym okresie czasu?) zadany w postaci kontekstowej bazy wiedzy. Dane jest środowisko programowania z ograniczeniami o znanych możliwościach funkcjonalnych i wymaganiach technicznych (np. **Oz Mozart**, **Ilog**, itd.). Problem sprowadza się do odpowiedzi na pytanie:

Czy istnieje metoda budowy systemów wspomagania decyzji pozwalających odpowiadać na określone zestawy pytań rutynowych, w kontekście posiadanej wiedzy określonego obiektu, w trybie interakcyjnym?

Przedstawione powyżej pytanie determinuje konieczność odpowiedzi na pytania cząstkowe:

- *Czy istnieje specyfikacja obiektu w postaci zbioru ograniczeń (faktów) i zmiennych decyzyjnych gwarantująca istnienie niepustej przestrzeni potencjalnych rozwiązań?*
- *Czy istnieje czasowo efektywna strategia poszukiwania rozwiązań dopuszczalnych w niepustej przestrzeni potencjalnych rozwiązań?*

Odpowiedź na powyższe pytania umożliwia budowę systemów interakcyjnego wspomagania decyzji.

1.4. Podsumowanie

Rozważany w pracy problem sprowadza się do poszukiwania sposobu budowy zadaniowo zorientowanych systemów interakcyjnego wspomagania decyzji bazujących na technikach programowania z ograniczeniami *CP* i metodzie logiczno-algebricznej. Wspomaganie decyzji polega na udzielaniu przez system odpowiedzi na zadawane przez decydenta pytania. Sprowadza się to zwykle do konieczności rozwiązania szeregu problemów (w tym przypadku problemów kombinatorycznych). W rozważanych systemach poszukiwane są rozwiązania dopuszczalne spełniające wszystkie ograniczenia wynikające z właściwości obiektu i żądań użytkownika. Poszukiwanie rozwiązania dopuszczalnego w takich

problemach, jak problemy marszrutowania, harmonogramowania, itp., wiąże się z koniecznością rozwiązania problemu o dużej złożoności obliczeniowej. Ze względu na rozmiar przestrzeni potencjalnych rozwiązań wymagane jest by systemy wspomagania decyzji posiadały informację o warunkach gwarantujących istnienie w tej przestrzeni rozwiązań dopuszczalnych. Ponadto interakcyjność systemu wymusza posiadanie efektywnych czasowo strategii przeszukiwaniu przestrzeni potencjalnych rozwiązań.

Wykorzystanie metody logiczno-algebraicznej do budowy reprezentacji wiedzy, charakteryzującej obiekt, pozwala na poszukiwanie określonych właściwości rozważanego problemu (w tym przypadku, warunków wystarczających). Wykorzystywane w tym celu mechanizmy wnioskowania można przenieść na płaszczyznę technik programowania z ograniczeniami i w tej nowej reprezentacji, poprzez rozwiązanie odpowiedniego problemu *PSO*, poszukiwać warunków wystarczających oraz samego rozwiązania dopuszczalnego.

W przeprowadzonych dalej badaniach skoncentrowano się zatem na ocenie możliwości wykorzystania metody logiczno-algebraicznej implementowanej w technikach programowania z ograniczeniami do budowy systemów wspomagania decyzji. W tym celu wykorzystano język programowania z ograniczeniami **Oz Mozart**.

2. Systemy interakcyjnego wspomagania decyzji

Zgodnie z ideą przedstawioną na rysunku 1.1 systemy wspomagania decyzji stanowią narzędzia, celem których jest udzielanie odpowiedzi na pytania stawiane przez użytkownika w kontekście rozważanego obiektu. Dla przykładu obiektem może być małe i średnie przedsiębiorstwo charakteryzowane przez takie parametry jak: dostępność zasobów produkcyjnych, ich liczba, dostępna struktura transportowa, zdolność produkcyjna, pojemność magazynów składowania międzyoperacyjnego, itp. Inne obiekty (np. systemy transportowe, sieci komputerowe, itp.) opisywane są przez odpowiednie charakteryzujące je parametry. Pytania stawiane przez użytkownika w kontekście małych i średnich przedsiębiorstw dotyczą m.in.:

- bilansowania możliwości producenta (systemu wytwórczego) i potrzeb klienta (zlecenia produkcyjnego)
- bilansowania dostępnej liczby wózków transportowych, ich pojemności oraz nominalnych prędkości z liczbą, rozmieszczeniem i pojemnościami dostępnych magazynów składowania międzyoperacyjnego,
- możliwości wykonania planowanego zlecenia produkcyjnego w założonym terminie,
- możliwości podjęcia realizacji nowego zlecenia w kontekście dostępnych zdolności produkcyjnych, w zadanym horyzoncie czasu.

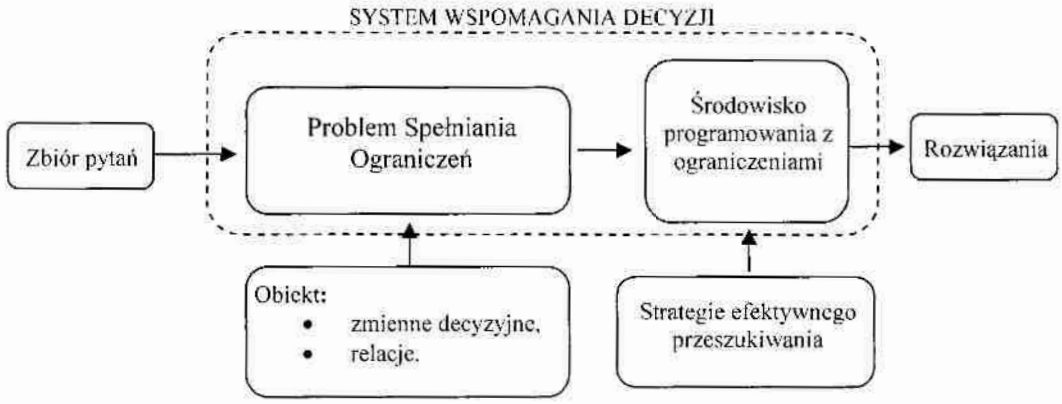
Przedstawiona grupa zagadnień stanowi tylko przykładowy fragment zakresu jakiego mogą dotyczyć pytania stawiane przez użytkownika. Biorąc pod uwagę różnorodność i liczbę potencjalnych pytań, jakie użytkownik może sformułować w stosunku do systemu wspomagania decyzji, oczywistym jest, że budowa systemów będących w stanie udzielić odpowiedzi na każde z możliwych pytań nie jest możliwa. Systemy wspomagania decyzji zwykle ograniczają się w swoim działaniu tylko do określonych klas obiektów i umożliwiają udzielanie odpowiedzi tylko w obszarze arbitralnie zadanej grupy pytań rutynowych. Innymi słowy, są to systemy zorientowane zadaniowo czyli budowane pod kątem określonych, znanych cech, właściwości i parametrów obiektu. Ze względu na specyfikę warunków pracy tego typu systemów, stawiane są im następujące wymagania [6], [7] :

- system powinien gwarantować istnienie odpowiedzi na zadany zbiór pytań rutynowych,
- system powinien wyznaczać odpowiedź na zadane pytanie w trybie *on-line*, umożliwiającym interakcyjną współpracę z użytkownikiem.

Interakcyjność większości dostępnych obecnie na rynku systemów wspomagania decyzji jest gwarantowana przez stosowanie efektywnych strategii przeszukiwania przestrzeni potencjalnych rozwiązań. Struktura takiego systemu oparta na technikach programowania z ograniczeniami została przedstawiona na rysunku 2.2.

Zmienne decyzyjne i właściwości obiektu (wyrażone w postaci relacji) formułowane są zwykle w postaci problemu *PSO*, który szerzej omawiany jest w rozdziale 3. Problem spełniania ograniczeń, specyfikowany przez zbiór zmiennych decyzyjnych, zbiór ich dziedzin

oraz zbiór ograniczeń, jest rozwiązywany w środowiskach programowania z ograniczeniami (**Oz Mozart, Ilog, Eclipse**, itp.), w których stosowane są efektywne strategie poszukiwania rozwiązań dopuszczalnych [29], [53], [81]. Postać *PSO* formułowana jest w taki sposób, by rozwiązanie problemu stanowiło jednocześnie odpowiedź na zadane przez użytkownika pytanie rutynowe. W ogólności, jednemu pytaniu odpowiada jedna postać *PSO*.



Rys. 2.2. Struktura sytemu wspomagania decyzji

Okazuje się, że stosowanie efektywnych strategii przeszukiwania przestrzeni potencjalnych rozwiązań, opartych na procedurach propagacji ograniczeń oraz dystrybucji zmiennych, w wielu rzeczywistych problemach [17] nie daje gwarancji uzyskania rozwiązania w trybie interakcyjnym. Przestrzeń potencjalnych rozwiązań okazuje się zbyt duża i/lub nie zawiera rozwiązań dopuszczalnych. Przykład 2.1 przedstawia taką sytuację.

Przykład 2.1. Przeszukiwanie przestrzeni dla ograniczeń nieliniowych

Celem przykładu jest ilustracja problemu, w którym poszukiwanie rozwiązań dopuszczanych prowadzi do przeglądu zupełnego.

Dane są zmienne x_1, x_2, x_3, x_4 oraz dziedziny tych zmiennych $D_{x,1} = D_{x,2} = D_{x,3} = D_{x,4} = \{1,..,1000\}$, $x_1 \in D_{x,1}$, $x_2 \in D_{x,2}$, $x_3 \in D_{x,3}$, $x_4 \in D_{x,4}$. Dane są ograniczenia C_1: $x_1^{x4} + x_2^{x4} = x_3^{x4}$, C_2: $x_4 = 3$. Należy odpowiedzieć na pytanie: *Czy istnieje taka kombinacja wartości zmiennych decyzyjnych* x_1, x_2, x_3, x_4, *dla których spełnione są ograniczenia* C_1, C_2? *Jeśli tak, to jaką ma postać ?*

W celu odpowiedzi na to pytanie formułowany jest problem *PSO*, w skład którego wchodzą zmienne x_1, x_2, x_3, x_4, ich dziedziny $D_{x,1}$, $D_{x,2}$, $D_{x,3}$, $D_{x,4}$ oraz ograniczenia C_1, C_2. Okazuje się, że ograniczenie C_1 stanowi przykład nieliniowego równania diofantycznego, dla którego (gdy $x_4 = 3$) nie istnieje rozwiązanie w zbiorze liczb całkowitych różnych od zera (twierdzenie Fermata). Bez względu na to jak duże będą dziedziny zmiennych decyzyjnych, żadna kombinacja wartości zmiennych x_1, x_2, x_3, x_4, nie spełnia ograniczenia C_1.

Procedura poszukiwania odpowiedzi na postawione pytanie, przez system odpowiadający strukturze z rysunku 2.2, jest następująca:

W pierwszej kolejności dla sformułowanego *PSO* określany jest rozmiar przestrzeni potencjalnych rozwiązań R_D: $R_D = ||D_{x,1}|| \cdot ||D_{x,2}|| \cdot ||D_{x,3}|| \cdot ||D_{x,4}||$, gdzie: $||D_{x,i}||$ oznacza liczbę

elementów zbioru $D_{x,1}$. Ograniczenie C_2 powoduje zawężenie dziedziny zmiennej x_4 do zbioru jednoelementowego: $||D_{x,4}|| = 1$. Otrzymana przestrzeń ma rozmiar $R_D = 10^9$. Ze względu na to, że ograniczenie C_1 ma charakter nieliniowy, przestrzeń nie jest dalej ograniczana [17], [53] (jest to wynikiem stosowanych procedur propagacji które „nie radzą sobie" z ograniczeniami nieliniowymi). System nie jest „świadomy" tego, że dla zadanego ograniczenia C_1 nie istnieje rozwiązanie dopuszczalne, dlatego też rozpoczyna systematyczne przeszukiwanie przestrzeni potencjalnych rozwiązań. Bez względu na to jaki rodzaj strategii zostanie użyty, konieczne jest przeszukanie całej przestrzeni potencjalnych rozwiązań. Przykładowo, zwiększanie dziedzin zmiennych prowadzi w konsekwencji do osiągnięcia takiego rozmiaru przestrzeni, że niemożliwe jest przeszukanie jej w akceptowalnym przez użytkownika czasie.

■

Przykład 2.1 ilustruje przypadek, w którym stosowanie efektywnych strategii przeszukiwania przestrzeni potencjalnych rozwiązań, nie gwarantuje udzielenia odpowiedzi w trybie interakcyjnym. Ponadto, poza odpowiedzią, że rozwiązanie nie istnieje, użytkownik nie dostaje żadnej innej informacji, nie wie na przykład, dlaczego rozwiązanie nie zostało uzyskane. Tego typu sytuacje pojawiają się w szczególności (choć nie tylko) gdy problem *PSO* jest opisywany ograniczeniami o charakterze nieliniowym [53]. Z tego też względu większość dostępnych obecnie na rynku systemów wspomagania decyzji bazujących na technikach programowania z ograniczeniami tylko w ograniczonym stopniu spełnia wymagania stawiane systemom interakcyjnego wspomagania decyzji. Systemy te dostarczają użytkownikowi narzędzia wykorzystujące szereg efektywnych strategii przeszukiwania jednak nie gwarantujących istnienia rozwiązań. Dlatego też wymaga się od tego typu systemów posiadania wiedzy na temat warunków gwarantujących istnienie odpowiedzi na zadane pytanie. Warunki tego typu dają gwarancję, że otrzymana przestrzeń potencjalnych rozwiązań (rozwiązań dopuszczalnych) jest przestrzenią niepustą.

Przykład 2.2. Przeszukiwanie przestrzeni z warunkami wystarczającymi

Dany jest problem z przykładu 2.1. Zgodnie z twierdzeniem Fermata równanie odpowiadające ograniczeniu C_1: $x_1^{x4} + x_2^{x4} = x_3^{x4}$ posiada rozwiązanie w zbiorze liczb całkowitych różnych od zera wtedy tylko, gdy C_3: $x_4 \leq 2$. Zatem wyrażenie $x_4 \leq 2$ stanowi warunek wystarczający, spełnienie którego gwarantuje, że rozwiązywany problem ma rozwiązanie. Uzupełnienie problemu *PSO* o ograniczenie C_3 umożliwia uzyskanie przestrzeni potencjalnych rozwiązań zawierającej zawsze rozwiązanie dopuszczalne. Jeśli zatem, jak poprzednio C_2: $x_4 = 3$, wówczas ograniczenie to jest sprzeczne z warunkiem wystarczającym C_3: $x_4 \leq 2$. Warunek wystarczający nie jest spełniony, zatem od razu na samym wstępie wiadomo, że rozwiązanie nie istnieje. Unika się w ten sposób zbędnego przeszukiwania przestrzeni potencjalnych rozwiązań. W przypadku spełnienia warunku C_3: $x_4 \leq 2$, w otrzymanej przestrzeni, istnieją rozwiązania dopuszczalne.

■

Przykład 2.2 pokazuje, że dla określonego systemu wspomagania decyzji, poza posiadaniem odpowiedniej strategii przeszukiwania, istotnego znaczenia nabiera wyznaczenie warunków wystarczających, spełnienie których gwarantuje istnienie odpowiedzi na zadane pytanie.

2.1. Systemy komputerowo zintegrowanego zarządzania

Struktura przedstawiona na rysunku 2.2 odpowiada systemom bazującym na technikach programowania z ograniczeniami. W rzeczywistości, istnieje szereg różnych podejść do budowy systemów wspomagania decyzji. Jednak, podobnie jak w przypadku technik programowania z ograniczeniami, dostępne systemy komputerowe nie spełniają wymagania pracy w trybie interakcyjnym. W szczególności, implementowane w nich rozwiązania nie uwzględniają specyfiki istniejących problemów (np. problemów składających się z podproblemów opisanych przez różne modele i wyrażone w różnych terminologiach). Przykład takiego pakietu stanowi profesjonalny program **Lingo** [57], [100], [74], (wykorzystujący metody programowania matematycznego), **Taylor** [84], [102], (implementujący metody symulacji komputerowej) czy na przykład **OptiTrans** [98]. Stosowanie pierwszego z nich w problemach decyzyjnych wiąże się z długim czasem oczekiwania na wynik. Złożony jest również sposób specyfikowania samego problemu. Stosowanie drugiego z wymienionych pakietów oprogramowania wymaga przygotowywania i przeprowadzania czasochłonnych i pracochłonnych eksperymentów. Dodatkowo, wypracowane decyzje ograniczają się do wcześniej symulowanych wariantów, a każdorazowa zmiana parametrów i cech rozpatrywanego obiektu, wiąże się z koniecznością ponawiania eksperymentu komputerowego [54], [72], [73]. Trzeci z wymienionych pakietów – **OptiTrans,** jest przykładem pakietu wspomagania decyzji z zakresu planowania transportu. Mimo licznych modułów (mapy cyfrowe, rozkład załadunku, szczegółowy plan przewozów i szereg innych) nie uwzględnia w pełni planowania dystrybucji dokładnie na czas (nie bilansuje potrzeb przewozowych w każdej jednostce czasu horyzontu planowania) [53]. Ponadto, obsługa tak złożonego systemu wymaga wykwalifikowanego personelu oraz odpowiedniej platformy sprzętowej [98]. Dodatkowo, w wielu przypadkach dostępne rozwiązania wykorzystywane w systemach wspomagania przerastają możliwości finansowe użytkownika systemu.

Pakiet **OptiTrans** jest przykładem na to, że budowa narzędzi o uniwersalnym charakterze w rzeczywistości ogranicza ich wykorzystanie tylko do rozwiązywania problemów ogólnych (to znaczy takich, gdzie nie ma możliwości wprowadzania własnych ograniczeń charakterystycznych dla spotkanego przez użytkownika przypadku). Uniwersalność problemu określa jak wiele wariantów (odmian) danego problemu można rozwiązywać przy użyciu dostępnego narzędzia. Rozwiązywane problemy charakteryzują się zwykle ograniczonym stopniem szczegółowości specyfikacji. Stopień szczegółowości określa jak wiele zmiennych i zależności charakterystycznych dla danego problemu zostało

uwzględnionych w jego specyfikacji. Oznacza to, że wraz ze wzrostem poziomu uniwersalności systemu wspomagania decyzji, maleje stopień szczegółowości rozwiązywanych problemów. Wynika to bezpośrednio z faktu, że im stopień szczegółowości specyfikacji problemu jest większy, tym bardziej niezbędne staje się uwzględnianie coraz większej ilości zmiennych i specyficznych zależności istotnych tylko dla indywidualnych przypadków spotykanych problemów. Różnorodność postaci rozwiązywanych problemów wymusza zatem konieczność stosowania wielu dedykowanych (oddzielnych dla każdej postaci problemu) metod rozwiązania. Oznacza to, że wzrost uniwersalności systemu implikuje wzrost jego kosztu. Stąd osiągnięcie określonego poziomu szczegółowości problemu z jednej strony oznacza brak ekonomicznego uzasadnienia dla budowy uniwersalnych systemów wspomagania decyzji, z drugiej zaś postuluje konieczność wdrażania systemów zorientowanych zadaniowo, projektowanych pod kątem potrzeb i struktury obiektu, gwarantujących interakcyjny tryb wspomagania decyzji.

2.2. Systemy ekspertowe

W odróżnieniu od przedstawionych powyżej systemów wspomagania decyzji systemy ekspertowe stanowią narzędzia, w których można wyróżnić bazę wiedzy, zawierającą wiedzę dziedzinową istotną dla podejmowania decyzji, oraz odrębny moduł wnioskujący, korzystający z bazy wiedzy dla wypracowania tych decyzji. Taki podział stanowi cechę charakterystyczną tego typu systemów i jednocześnie stanowi ich podstawową zaletę. Wydzielenie bazy wiedzy jako odrębnej części systemu daje możliwość swobodnego dokonywania zmian (np. uaktualnień, zmiany podstawowych parametrów, modyfikacji określonych funkcji) przez użytkownika systemu bez konieczność ingerencji w jego strukturę. Z tej perspektywy, od użytkownika wymagane jest tylko sformułowanie wiedzy zgodnie z przyjętym w danym systemie formalizmem reprezentacji wiedzy.

Wspomaganie/podejmowanie decyzji przez systemy ekspertowe polega na przetwarzaniu dostarczonej wiedzy pod kątem uzyskania odpowiedzi na zadane przez użytkownika pytanie. Pomijając aspekty poprawności i „jakości" dostarczonej wiedzy, można wyróżnić dwie wzajemnie zależne cechy decydujące o funkcjonalności systemu ekspertowego. Pierwsza cecha określa sposób reprezentacji bazy wiedzy. Powszechnie wykorzystywane są regułowe bazy oparte o logikę rachunku zdań, lub rachunku predykatów [62], [58], [12], [13]. Znane są również reprezentacje wiedzy za pomocą list, ram czy sieci semantycznych [58], [11]. Współczesne systemy ekspertowe budowane są przy użyciu głównie języków przetwarzania symbolicznego takich jak **Prolog** [62], [61], czy **Lisp** [41], wykorzystujących reprezentację wiedzy w postaci rachunku predykatów (**Prolog**), i list (**Lisp**). Drugą cechą wpływającą na funkcjonalność systemów jest sposób wnioskowania i przetwarzania wiedzy. Zauważalna jest zasada mówiąca, że im „swobodniejsza" gramatyka reprezentacji wiedzy tym bardziej złożone są mechanizmy wnioskowania. Jako przykład można przedstawić metodę logiczno-algebraiczną, która wykorzystuje do opisu wiedzy

formalizm w postaci zdań logicznych o dowolnej strukturze (nie ograniczonej tylko do postaci implikacji jak to ma miejsce w systemach regułowych).

Taki sposób reprezentacji wiedzy (bardzo wygodny z punktu widzenia od strony użytkownika) niesie jednak za sobą konieczność stosowania algorytmów wnioskowania charakteryzujących się złożonością wykładniczą. Okazuje się, że w wielu problemach uzyskanie odpowiedzi wymaga irracjonalnie długiego czasu – co jest sprzeczne z wymaganiem interakcyjności systemów wspomagania decyzji. Mechanizm wnioskowania stanowi zatem podstawowe ograniczenie użyteczności systemów ekspertowych opartych o tę metodę.

Innym przykładem są systemy ekspertowe budowane w oparciu o język programowania **Prolog**. W systemach tych przyjęte jest założenie zamkniętego świata [62]. Dla systemów ekspertowych zakłada się, że prawdą jest tylko to co wynika z reguł i faktów zadeklarowanych przez użytkownika. Innymi słowy jeżeli czegoś nie można się wywnioskować z reguł i faktów bazy wiedzy, uważa się to za nieprawdę. Założenie to uniemożliwia udzielenie odpowiedzi typu „nie wiem", przez co użytkownik nie jest w stanie określić czy wprowadzona przez niego wiedza jest wystarczająca do rozwiązania określonego problemu, co znacznie utrudnia proces rozbudowy bazy wiedzy. Pomimo tej wady stosowanie przyjętego założenia powoduje uproszczenie podstawowych operatorów logicznych. Do opisu wiedzy wykorzystuje się reguły w postaci klauzul Horna [11], implikacja logiczna zastępowana jest implikacją regułową, która jest prawdziwa tylko w przypadku gdy z prawdy wynika prawda lub z fałszu wynika fałsz. Ponadto stosowana składania języka (**Prolog**) pozwala uniknąć często niewygodnych dla komputerowego przetwarzania reguł wykorzystujących operator dyzjunkcji („$\vee$"). Powyższe ograniczenia przekładają się na efektywne mechanizmy wnioskowania oparte na zasadzie rezolucji (a w szczególności na zasadzie *modus ponens*). Tego typu podejście pozwala na przetwarzanie wiedzy o rozmiarach znacznie większych niż w przypadku przedstawionej poprzednio metody logiczno-algebraicznej.

Systemy tego typu są w stanie udzielać odpowiedzi w trybie na bieżąco. Mimo licznych zastosowań tego typu systemów ekspertowych w różnych obszarach działalności człowieka, istnieją dziedziny, dla których wymagany opis wiedzy znacznie przekracza ich możliwości. Przykładem takich problemów, są problemy harmonogramowania występujące w systemach sterowania operacyjnego. Oczywiście istnieją mechanizmy przetwarzania dowolnej struktury zdań logicznych do postaci klauzul Horna jednak w wielu przypadkach charakteryzują się one złożonością ekspotencjalną, co wiąże się z wydłużeniem czasu i zagrożeniem utraty „interakcyjności" systemu ekspertowego.

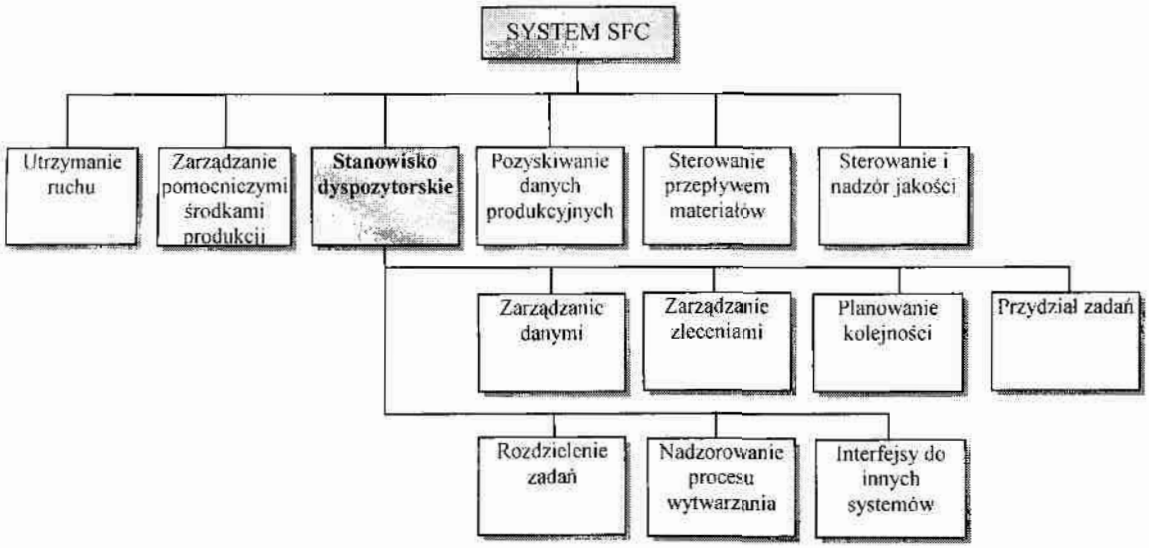
Obecnie silnie determinowana jest potrzeba swobodnej modyfikacji wiedzy od strony użytkownika. Wymaganie to wymusza potrzebę wyrażania wiedzy w postaci zdań o dowolnej strukturze. Niewystarczające okazują się zatem systemy oparte przykładowo na klauzulach Horna. Z kolei, stosowanie reprezentacji wiedzy o rozbudowanej i złożonej gramatyce wymaga użycia odpowiednich efektywnych czasowo metod wnioskowania. Istnieje zatem potrzeba poszukiwania strategii efektywnego przetwarzania wiedzy.

2.3. Systemy sterowania operacyjnego

Szczególną grupę systemów wspomagania decyzji stanowią systemy wykorzystujące techniki sterowania operacyjnego (ang. Shop Floor Control, *SFC*). Znajdują one głównie zastosowanie w przedsiębiorstwach produkcyjnych podczas realizacji zleceń na poziomie operacyjnym tzn. na wydziale produkcyjnym. Celem systemów wykorzystujących techniki *SFC* jest koordynacja technicznego i organizacyjnego przepływu informacji wewnątrz przedsiębiorstwa (np. wytwórczego) w ramach krótkoterminowego zarządzania produkcją. Stanowią one zorientowane zadaniowo narzędzia wspomagania decyzji, których zadaniem jest zbliżenie przebiegu określonego procesu do optimum przy często sprzecznych ze sobą założeniach. Przykładowe założenia dotyczą m.in.:

- dotrzymania terminów zamówień,
- minimalizacji czasów realizacji zleceń,
- maksymalnego wykorzystania zasobów produkcyjnych,
- minimalizacji kosztów.

Systemy *SFC* charakteryzują się strukturą modułową, w której centralną funkcję pełni moduł stanowiska dyspozytorskiego (rysunek 2.3). Odpowiedzialny jest on zazwyczaj za ustalenie kolejności realizacji zleceń oraz za koordynowanie i nadzorowanie zasobów produkcyjnych, pod względem ich czasowej i ilościowej dyspozycyjności.



Rys. 2.3. Struktura systemu SFC - przykładowe moduły funkcjonalne [34]

Stanowisko dyspozytorskie zleca pozostałym modułom (jeśli takie istnieją), wykonanie dodatkowych czynności z zakresu przepływu informacji lub procesu planowania. Każdemu modułowi odpowiada odrębna funkcja. Podział systemu na moduły ma na celu zredukowanie złożoności procesu produkcyjnego. Struktura modułowa powinna zapewnić szybką i elastyczną reakcję systemu na zmiany zachodzące w przedsiębiorstwie, wymagania użytkowników itp.

Wspomaganie decyzji w systemach *SFC* polega na dostarczeniu, tak szeregu narzędzi wspierających pracę planisty przy opracowaniu harmonogramów, jak i narzędzi umożliwiających automatyczne generowanie gotowych harmonogramów spełniających żądania użytkownika. Wykorzystywane w tym celu metody są głównie metodami heurystycznymi, dedykowanymi dla ściśle określonych problemów.

Nadal największą popularnością cieszą się nadal reguły priorytetu [34], które są stosowane do wyboru kolejności oczekujących do stanowiska procesów technologicznych. W celu wyboru reguł priorytetu wykorzystywane są badania symulacyjne, rzadziej stosowane są metody aproksymacyjne bądź w przypadku problemów optymalizacyjnych, algorytmy genetyczne [34]. W ostatnich latach do planowania harmonogramów wykorzystywane są również techniki oparte na algebrze (max, +) [50], oraz techniki programowania z ograniczeniami *CP*.

Stosowanie metod heurystycznych stanowi podstawową wadę systemów *SFC*. Producenci systemów *SFC* umieszczają w nich ogromną ilości różnych funkcji przeznaczonych do rozwiązywania szczególnych wariantów spotykanych problemów. Tak bardzo rozbudowana funkcjonalność systemów w znacznym stopniu ogranicza ich elastyczność, a także utrudnia sprawną obsługę. Ponadto, należy zaznaczyć, że stale rozwijana funkcjonalność nie odpowiada rzeczywistym wymaganiom użytkowników. Systemy te pozbawione są możliwości modyfikacji ich podstawowych funkcji [34].

Podsumowując, systemy *SFC* są przykładem, w którym elastyczność wynikająca ze struktury modułowej systemu, ograniczana jest w wyniku zbyt dużej złożoności zaimplementowanych mechanizmów planowania harmonogramów. Oznacza to, potrzebę poszukiwania metod ujednolicających procedury rozwiązywania problemów. Inaczej, mówiąc, wymagane jest posiadanie metod deklaratywnych.

2.4. Podsumowanie

Z przedstawionej charakterystyki przykładowych systemów wspomagania decyzji wyraźnie widać, że brak jest rozwiązań gwarantujących spełnienie wymagań interakcyjności systemów. Techniki programowania z ograniczeniami nie gwarantują istnienia rozwiązań dopuszczalnych, systemy ekspertowe nie dysponują efektywnymi mechanizmami wnioskowania, w systemach sterowania operacyjnego brak jest jednolitego mechanizmu rozwiązywania problemów.

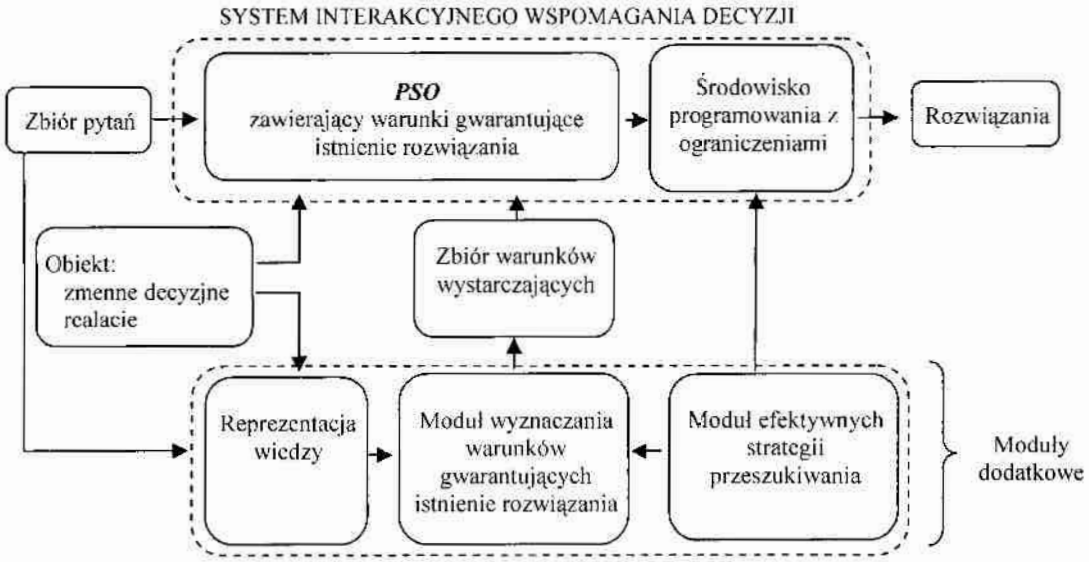
Analizując powyższe potrzeby scharakteryzowane zostały ogólne cechy jakie powinny spełniać systemy interakcyjnego wspomagania decyzji:

- systemy dedykowane do wspomagania decyzji tylko dla określonej klasy obiektów,
- mechanizmy wyznaczania odpowiedzi bazujące na technikach programowania z ograniczeniami,
- reprezentacja wiedza charakteryzująca obiekt odseparowana od pozostałych części systemu,

- specyfikacja problemu, w postaci zbioru ograniczeń, pozyskiwana jest z reprezentacji wiedzy (uzyskiwana specyfikacja zawiera warunki wystarczające gwarantujące istnienie odpowiedzi na zadane pytanie),
- struktura systemu ma charakter modułowy.

W konsekwencji zaproponowana została struktura systemu interakcyjnego wspomagania decyzji (rysunek 2.4) zawierająca **moduł wyznaczania warunków wystarczających** oraz zmodyfikowany **moduł efektywnych strategii przeszukiwania**. **moduł wyznaczania warunków wystarczających** jest odpowiedzialny za wyznaczenie dla zadanego zbioru pytań rutynowych, warunków, które w kontekście właściwości opisywanego obiektu, gwarantują istnienie odpowiedzi. **Moduł efektywnych strategii przeszukiwania**, oprócz strategii rozwiązania *PSO*, określa strategie poszukiwania tych warunków.

Systemy odpowiadające tego typu strukturze, powinny spełniać wymagania stawiane systemom interakcyjnego wspomagania decyzji – udzielają odpowiedzi w trybie na bieżąco oraz gwarantują istnienie rozwiązań na zadany zbiór pytań rutynowych.



Rys. 2.4. Struktura interakcyjnego sytemu wspomagania decyzji

Odpowiedzi na zadane pytania otrzymywane są w wyniku rozwiązania odpowiedniego problemu *PSO*. Specyfikacja problemu zawiera warunki wystarczające gwarantujące istnienie odpowiedzi na zadane pytanie. Problem rozwiązany jest przy użyciu efektywnych czasowo (dedykowanych) strategii przeszukiwania. Zbiór warunków wystarczających wyznaczany jest z reprezentacji wiedzy poprzez badanie jej spójności (**moduł wyznaczania warunków gwarantujących istnienie rozwiązania**). Sama reprezentacja wiedzy stanowi opis obiektu w przyjętej terminologii.

W kontekście przyjętej struktury, projektowanie systemów interakcyjnego wspomagania decyzji wymaga zatem analizy następujących zagadnień:

- sposobu modelowania reprezentacji wiedzy,
- weryfikacji bazy wiedzy,
- efektywnych strategii wyznaczania rozwiązań.

3. Model reprezentacji bazy wiedzy

3.1. Metoda logiczno-algebraiczna

Metoda logiczno-algebraincza wykorzystywana jest do rozwiązywania zadań w systemach ekspertowych z reprezentacją wiedzy w postaci faktów. W rozważanym przypadku reprezentację wiedzy określa się jako zbiór faktów i reguł reprezentujących właściwości i zależności charakterystyczne dla opisywanego obiektu i może być ona traktowana jako uogólnienie tradycyjnych modeli matematycznych [38].

3.1.1. Reprezentacja wiedzy

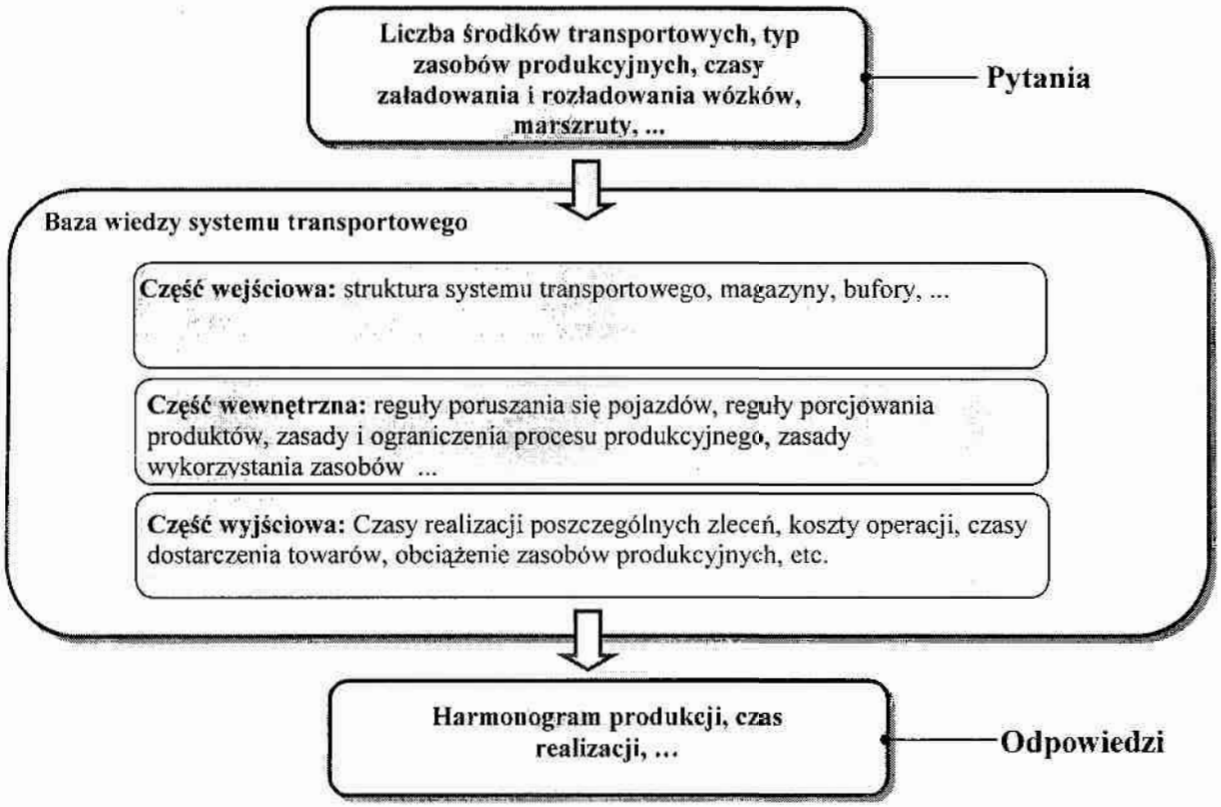
Obiekty, dla których obywa się wspomaganie decyzji postrzegane są w postaci bazy wiedzy stanowiącej zbiór wszystkich informacji na temat ich cech i właściwości. Rysunek 3.1 ilustruje przykład bazy wiedzy dla systemu produkcyjnego. W bazie wiedzy, wyróżnia się trzy części: wejściową, wyjściową i wewnętrzną.

Część wejściowa stanowi zbiór parametrów i właściwości opisujących podstawowe cechy obiektu, znane i zadawane przez użytkownika. W rozważanej bazie wiedzy są to zmienne dotyczące struktury systemu transportowego, zmienne określające pojemności magazynów, cechy środków transportu, itp. Użytkownik systemu wspomagania decyzji zadaje wartości liczbowe tych zmiennych określając w ten sposób wejściowy (aktualny) stan obiektu (systemu transportowego).

Część wyjściowa stanowi zbiór parametrów i właściwości opisujących te cechy obiektu, które nie są znane (bądź znane tylko częściowo) przez użytkownika systemu, a które chciałby poznać w pełni. Dla przykładu, w bazie wiedzy z rysunku 3.1, część wyjściową stanowią zmienne określające koszty realizowanych operacji, czasy dostarczenia towarów, obciążenie zasobów produkcyjnych, itp. Użytkownik zwykle chce znać wartości oraz cechy tych parametrów w zależności od zadanych wartości i właściwości parametrów wejściowych obiektu.

Część wewnętrzna stanowi najistotniejszą część bazy wiedzy, jest pomostem między częścią wejściową i wyjściową. Określa zbiór ogólnych zasad, które obowiązują w rozważanym obiekcie. Dla przykładu, w przedsiębiorstwach produkcyjnych są to zasady poruszania się środków transportu, zasady składowania, zasady wynikające z ograniczeń procesu produkcyjnego.

Podział bazy wiedzy na części dokonywany jest w sposób arbitralny przez użytkownika. Zmienne, które według jednego użytkownika są uznane za wejściowe, przez innego mogą być uznane za pomocnicze bądź wyjściowe. Niemniej jednak, podział bazy wiedzy umożliwia formułowanie, w kontekście posiadanej wiedzy, pytań. Zwykle przyjmuje się, że z częścią wejściową utożsamiane są pytania użytkownika, a z częścią wyjściową odpowiedzi.



Rys. 3.1. Baza wiedzy dla systemu produkcyjnego

Struktura pytania ma postać: *Jaką postać ma część wyjściowa bazy wiedzy (wartości parametrów wyjściowych np. harmonogramy pracy), gdy zadana jest określona postać części wejściowej bazy wiedzy (wartości parametrów wejściowych np. liczba wózków, pojemność środków transportu, itp.)?* Pytanie może mieć też postać odwrotną: *Jaką postać ma część wejściowa bazy wiedzy przy znanej postaci części wyjściowej?* Poszukiwanie odpowiedzi na tego typu pytania determinuje potrzebę wyrażenia wiedzy przy użyciu odpowiedniego formalizmu.

Przyjęto, że baza wiedzy opisująca określony obiekt jest wyrażana w terminologii metody logiczno-algebraicznej. Zgodnie z metodą logiczno-algebraiczną wiedza na temat obiektu, opisywana jest w postaci reprezentacji wiedzy [25], [26], [38]:

$$KB = <\alpha, F(\alpha)>, \tag{3.1}$$

gdzie: $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ – jest ciągiem formuł elementarnych opisujących określone cechy obiektu; α_i – i-ta formuła elementarna, $a_i = w(\alpha_i) \in \{0,1\}$ – logiczna wartość formuły α_i. $F(\alpha) = \{F_1(\alpha), F_2(\alpha), \ldots, F_K(\alpha)\}$ – zbiór faktów opisujących relacje pomiędzy poszczególnymi formułami elementarnymi α na poziomie zdań logicznych (używając operacji: koniunkcji, alternatywy, negacji i implikacji); $Q_j(a)$ – oznacza wartość logiczną zdania $F_j(\alpha)$.

Ciągowi $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ odpowiada ciąg wartości logicznych $a = (a_1, a_2, \ldots, a_N)$.

W reprezentacji wiedzy $KB = <\alpha, F(\alpha)>$, analogicznie do bazy wiedzy z rysunku 3.1,

wyróżnia się ciągi:

$\alpha u = (\alpha u_1, \alpha u_2, \ldots, \alpha u_k)$ – wejściowych formuł elementarnych opisujących właściwości i zmienne wejściowe systemu, formuły αu stanowią część wejściową bazy wiedzy, αu_i są elementami ciągu α,

$\alpha y = (\alpha y_1, \alpha y_2, \ldots, \alpha y_p)$ – wyjściowych formuł elementarnych opisujących właściwości wyjściowe systemu, formuły αy stanowią część wyjściową bazy wiedzy, αy_i są elementami ciągu α,

$\alpha w = (\alpha w_1, \alpha w_2, \ldots, \alpha w_r)$ – pomocniczych formuł elementarnych, formuły αw, stanowią część wewnętrzną bazy wiedzy, αw_i są elementami ciągu α.

Ciągom αu, αy, αu odpowiadają ciągi wartości logicznych: $au = (au_1, au_2, \ldots, au_k)$, $ay = (ay_1, ay_2, \ldots, ay_p)$, $aw = (aw_1, aw_2, \ldots, aw_r)$.

Relacje opisujące związki pomiędzy formułami wejściowymi αu i wyjściowymi αy opisują zbiory:

$Fu(\alpha u) = \{Fu_1(\alpha u), Fu_2(\alpha u), \ldots, Fu_{P_f}(\alpha u)\}$ – faktów wejściowych, opisujących związki między formułami αu,

$Fy(\alpha y) = \{Fy_1(\alpha y), Fy_2(\alpha y), \ldots, Fy_{R_f}(\alpha y)\}$ – faktów wyjściowych, opisujących związki między formułami αy.

Zbiorom $Fu(\alpha u)$, $Fy(\alpha y)$, odpowiadają ciągi wartości logicznych faktów: $Qu(au) = (Qu_1(au), Qu(au), \ldots, Qu_{P_f}(au))$, $Qy(ay) = (Qy_1(ay), Qy_2(ay), \ldots, Qy_{R_f}(ay))$, gdzie: $Qu_j(au)$, $Qy_j(ay)$ – określają kolejno wartości logiczne faktów $Fu_j(\alpha u)$, $Fy_j(\alpha y)$.

Przyjmuje się, że wszystkie fakty opisujące dany obiekt są prawdziwe:

$Q(a) = Q(au, aw, ay) = \bar{1}$, co oznacza: $Q_1(a) = 1$, $Q_2(a) = 1, \ldots, Q_K(a) = 1$,

$Qu(au) = \bar{1}$, co oznacza: $Qu_1(au) = 1$, $Qu_2(au) = 1, \ldots, Qu_{P_f}(au) = 1$,

$Qy(ay) = \bar{1}$, co oznacza: $Qy_1(ay) = 1$, $Qy_2(ay) = 1, \ldots, Qy_{R_f}(ay) = 1$.

W ogólnym przypadku, w formułach α mogą występować zmienne, które wpływają na postać określonych formuł elementarnych α_i. Niech ciągi $u = (u_1, u_2, \ldots, u_{ku})$, $w = (w_1, w_2, \ldots, w_{kw})$, $y = (y_1, y_2, \ldots, y_{ky})$ oznaczają kolejno zmienne: wejściową, wewnętrzną i wyjściową: $u_i \in U$, $w_i \in W$, $y_i \in Y$. Założono, że w formułach wejściowych αu występują tylko zmienne u, w formułach wyjściowych αy tylko zmienne y, w formułach pomocniczych αw zmienne u, w, y. Zatem formuły elementarne mają postać:

$$\alpha = \upsilon(u, w, y), \qquad (3.2)$$
$$\alpha u = \upsilon u(u), \quad \alpha w = \upsilon w(u,w,y), \quad \alpha y = \upsilon y(y),$$

gdzie: $\alpha u = \upsilon u(u) = (\upsilon u_1(u), \upsilon u_2(u), \ldots, \upsilon u_{ku}(u))$ jest ciągiem funkcji zdaniowych określających postać formuł elementarnych αu. Analogicznie $\alpha w = \upsilon w(u,w,y)$, $\alpha y = \upsilon y(y)$. Elementy ciągów αu, αw, αy, tworzą łącznie ciąg α. W konsekwencji ciągi wartości logicznych $au(u)$, $aw(u,w,y)$, $ay(y)$ są funkcjami odpowiednich zmiennych.

W kontekście zmiennych u, w, y baza wiedzy ma postać:

$$KB = <U, W, Y; Re>, \qquad (3.3)$$

gdzie: U, Y, W – zbiory określające dziedziny zmiennych u, y, w,

$Re = \{(u,w,y)$: $Q(au(u), aw(u,w,y), ay(y)) = \bar{1}\}$ – relacja będąca zbiorem wszystkich trójek (u,w,y), dla których fakty $F(\upsilon u(u), \upsilon w(u,w,y), \upsilon y(y))$ opisujące system są prawdziwe $Q(au(u), aw(u,w,y), ay(y)) = \bar{1}$ (wartości logiczne faktów są równe 1),

$aw(u,w,y)$ – ciąg wartości logicznych elementów ciągu $\upsilon w(u,w,y)$,

$au(u)$ – ciąg wartości logicznych elementów ciągu $\upsilon u(u)$,

$ay(y)$ – ciąg wartości logicznych elementów ciągu $\upsilon y(y)$,

$Q(au(u), aw(u,w,y), ay(y)) = (Q_1(au(u), aw(u,w,y), ay(y)), Q_2(au(u), aw(u,w,y), ay(y)), ..., Q_K(au(u), aw(u,w,y), ay(y)))$ – jest zestawem wartości logicznych faktów będących funkcjami zmiennych u, w, y. W dalszej części, w celu uproszczenia zapisu, ciągi wartości logicznych faktów, których postać jest zależna od zmiennych u, w, y będą przedstawiane w postaci: $Q(u,w,y) = (Q_1(u,w,y), Q_2(u,w,y), \ldots, Q_K(u,w,y))$. Podobnie zbiór faktów $F(\upsilon u(u), \upsilon w(u,w,y), \upsilon y(y)) = \{F_1(\upsilon u(u), \upsilon w(u,w,y), \upsilon y(y)), F_2(\upsilon u(u), \upsilon w(u,w,y), \upsilon y(y)), ..., F_K(\upsilon u(u), \upsilon w(u,w,y), \upsilon y(y))\}$, w formie skrótowej zapisywane będą w postaci: $F(u,w,y) = \{F_1(u,w,y), F_2(u,w,y), \ldots, F_K(u,w,y)\}$.

$Q(au(u), aw(u,w,y), ay(y)) = \bar{1}$ oznacza przyporządkowanie każdemu elementowi $Q_i(au(u), aw(u,w,y), ay(y)]$)ciągu $Q(au(u), aw(u,w,y), ay(y))$ wartości logicznej 1.

Postaci zbiorów faktów wejściowych i wyjściowych, określających właściwości odpowiadających im części bazy wiedzy, w przypadku formuł uzależnionych od zmiennych u, y mają odpowiednio postać: $Fu(\upsilon u(u)) = \{Fu_1(\upsilon u(u)), Fu_2(\upsilon u(u)), \ldots, Fu_{P_f}(\upsilon u(u))\}$, $Fy(\upsilon y(y)) = \{Fy_1(\upsilon y(y)), Fy_2(\upsilon y(y)), \ldots, Fy_{R_f}(\upsilon y(y))\}$. Odpowiadają im ciągi wartości logicznych faktów postaci: $Qu(au(u)) = (Qu_1(au(u)), Qu_2(au(u)), \ldots, Qu_{P_f}(au(u)))$, $Qy(ay(y)) = (Qy_1(ay(y)), Qy_2(ay(y)), \ldots, Qy_{R_f}(ay(y)))$.

W celu uproszczenia zapisu, w sytuacjach gdzie nie będzie to prowadzić do nieporozumień, biory faktów wejściowych i wyjściowych będą przedstawiane w postaci: $Fu(u) = \{Fu_1(u), Fu_2(u), \ldots, Fu_{P_f}(y)), Fy(y) = (Fy_1(y), Fy_2(y), \ldots, Fy_{R_f}(y)\}$ z kolei ciągi wartości logicznych tych faktów będą przedstawiane w postaci: $Qu(u) = (Qu_1(u), Qu_2(u), \ldots, Qu_{P_f}(y))$, $Qy(y) = (Qy_1(y), Qy_2(y), \ldots, Qy_{R_f}(y))$.

Przykład 3.1. Reprezentacja wiedzy dla równania kwadratowego

Przykład ma na celu ilustrację postaci reprezentacji wiedzy KB, opisującej liczbę pierwiastków w równaniu kwadratowym.

Dane jest równanie kwadratowe postaci: $sx^2 + bx + c = 0$, przyjęto, że $s \neq 0$. Jak powszechnie wiadomo równanie nie posiada pierwiastków rzeczywistych gdy: $\Delta < 0$, posiada jeden pierwiastek podwójny $x_1 = x_2 = \frac{-b}{2s}$, gdy: $\Delta = 0$, posiada dwa pierwiastki $x_1 = \frac{-b-\sqrt{\Delta}}{2s}, x_2 = \frac{-b+\sqrt{\Delta}}{2s}$ rzeczywiste, gdy: $\Delta > 0$.

Powyższą wiedzę, zgodnie z terminologią metody logiczno algebraicznej, sprowadza się do postaci formuł i faktów.

W kontekście zmiennych s, b, c, Δ, x_1, x_2, zdefiniowano formuły:

$\alpha = \upsilon(s,b,c,\Delta,x_1,x_2) = (\upsilon_1(s,b,c,\Delta,x_1,x_2), \upsilon_2(s,b,c,\Delta,x_1,x_2), \ldots, \upsilon_{10}(s,b,c,\Delta,x_1,x_2))$,

$\upsilon_1(s,b,c,\Delta,x_1,x_2)$: $\Delta = b^2 - 4sc$,

$\upsilon_2(s,b,c,\Delta,x_1,x_2)$: $s \neq 0$,

$\upsilon_3(s,b,c,\Delta,x_1,x_2)$: $\Delta < 0$,

$\upsilon_4(s,b,c,\Delta,x_1,x_2)$: $\Delta > 0$,

$\upsilon_5(s,b,c,\Delta,x_1,x_2)$: $x_1 \in \varnothing$,

$\upsilon_6(s,b,c,\Delta,x_1,x_2)$: $\Delta = 0$,

$\upsilon_7(s,b,c,\Delta,x_1,x_2)$: $x_1 = x_2 = \dfrac{-b}{2s}$,

$\upsilon_8(s,b,c,\Delta,x_1,x_2)$: $x_1 = \dfrac{-b-\sqrt{\Delta}}{2s}$,

$\upsilon_9(s,b,c,\Delta,x_1,x_2)$: $x_2 = \dfrac{-b+\sqrt{\Delta}}{2s}$,

$\upsilon_{10}(s,b,c,\Delta,x_1,x_2)$: $x_2 \in \varnothing$.

Zbiór faktów opisujących związki między formułami α ma postać:

$F(\upsilon(s,b,c,\Delta,x_1,x_2)) = (F_1(\upsilon(s,b,c,\Delta,x_1,x_2)), F_2(\upsilon(s,b,c,\Delta,x_1,x_2)), \ldots, F_5(\upsilon(s,b,c,\Delta,x_1,x_2)))$,

$F_1(\upsilon(s,b,c,\Delta,x_1,x_2))$: $\upsilon_1(s,b,c,\Delta,x_1,x_2)$,

$F_2(\upsilon(s,b,c,\Delta,x_1,x_2))$: $\upsilon_2(s,b,c,\Delta,x_1,x_2)$,

$F_3(\upsilon(s,b,c,\Delta,x_1,x_2))$: $\upsilon_3(s,b,c,\Delta,x_1,x_2) \Leftrightarrow \upsilon_5(s,b,c,\Delta,x_1,x_2) \wedge \upsilon_{10}(s,b,c,\Delta,x_1,x_2)$,

$F_4(\upsilon(s,b,c,\Delta,x_1,x_2))$: $\upsilon_6(s,b,c,\Delta,x_1,x_2) \Leftrightarrow \upsilon_7(s,b,c,\Delta,x_1,x_2)$,

$F_5(\upsilon(s,b,c,\Delta,x_1,x_2))$: $\upsilon_4(s,b,c,\Delta,x_1,x_2) \Leftrightarrow \upsilon_8(s,b,c,\Delta,x_1,x_2) \wedge \upsilon_9(s,b,c,\Delta,x_1,x_2)$,

gdzie: $a_i(s,b,c,\Delta,x_1,x_2) = w(\upsilon_i(s,b,c,\Delta,x_1,x_2))$ wartość logiczna formuły υ_i.

Zbiór faktów w uproszczonym (i czytelniejszym) zapisie ma postać:

$F(s,b,c,\Delta,x_1,x_2) = (F_1(s,b,c,\Delta,x_1,x_2), F_2(s,b,c,\Delta,x_1,x_2), \ldots, F_5(s,b,c,\Delta,x_1,x_2))$,

$F_1(s,b,c,\Delta,x_1,x_2)$: $\Delta = b^2 - 4sc$,

$F_2(s,b,c,\Delta,x_1,x_2)$: $s \neq 0$,

$F_3(s,b,c,\Delta,x_1,x_2)$: $(\Delta < 0) \Leftrightarrow (x_1 \in \varnothing) \wedge (x_2 \in \varnothing)$,

$F_4(s,b,c,\Delta,x_1,x_2)$: $(\Delta = 0) \Leftrightarrow \left(x_1 = x_2 = \dfrac{-b}{2s} \right)$,

$F_5(s,b,c,\Delta,x_1,x_2)$: $(\Delta > 0) \Leftrightarrow \left(x_1 = \dfrac{-b-\sqrt{\Delta}}{2s} \right) \wedge \left(x_2 = \dfrac{-b+\sqrt{\Delta}}{2s} \right)$.

W takiej notacji wartości logiczne faktów są funkcjami zmiennych s, b, c, Δ, x_1, x_2.

Relacja Re wchodząca w skład reprezentacji wiedzy ma postać:

$$Re = \{(s,b,c,\Delta,x_1,x_2): Q(s,b,c,\Delta,x_1,x_2) = \bar{1}\}.$$

gdzie: $Q(s,b,c,\Delta,x_1,x_2) = \bar{1}$ - oznacza skrótowy zapis wyrażenia $Q[a(s,b,c,\Delta,x_1,x_2)] = \bar{1}$.

Zgodnie z (3.3) reprezentacja wiedzy ma postać:

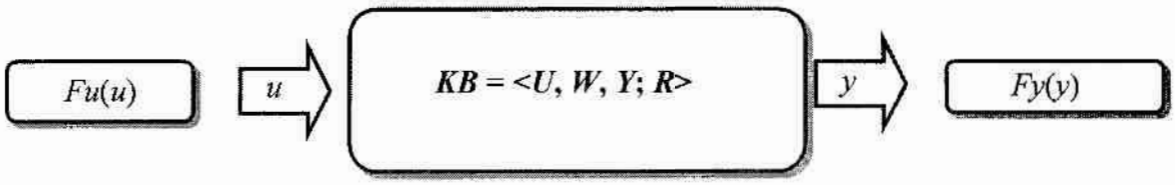
$$KB = \langle S, B, C, \Delta', X_1, X_2; Re \rangle,$$

gdzie: S, B, C, Δ', X_1, X_2 – zbiory wartości zmiennych s, b, c, Δ, x_1, x_2: $s \in S$, $b \in B$, $c \in C$, $\Delta \in \Delta'$, $x_1 \in X_1$, $x_2 \in X_2$.

Reprezentacja wiedzy KB stanowi więc reprezentację bazy wiedzy opisującej zasady określające liczbę pierwiastków równania kwadratowego. W powyższej reprezentacji nie wyszczególniono postaci zmiennych wejściowych, wyjściowych i pomocniczych.

■

Baza wiedzy z rysunku 3.1 może być postrzegana w kontekście przedstawionej terminologii jako układ wejście/wyjście (rysunek 3.2). Cała baza wiedzy - część wejściowa, wewnętrzna i wyjściowa - jest reprezentowana zależnością (3.3). Wejście i wyjście układu stanowią kolejno: zmienne wejściowe u, zmienne wyjściowe y. Właściwości wejścia i wyjścia układu są opisywane zbiorem faktów $Fu(u)$, $Fy(y)$. Właściwość $Fu(u)$ (lub $Fy(y)$) jest prawdziwa (spełniona) jeżeli wartości logiczne wszystkich faktów $Fu_i(u)$ (lub $Fy_i(y)$) są równe 1: $Qu_i(u) = 1$ dla $i = 1, 2, \ldots Pf$, ($Qy_i(u) = 1$ dla $i = 1, 2, \ldots Rf$).



Rys. 3.2. Reprezentacja wiedzy w postaci układu typu wejście/wyjście

W metodzie logiczno-algebraicznej, dla zadanej reprezentacji wiedzy KB, można wyróżnić trzy podstawowe rodzaje pytań [25], [38].

1) Jeżeli dla danego systemu prawdziwa jest właściwość wejściowa $Fu(u)$, to czy prawdziwa jest właściwość wyjściowa $Fy(y)$?
2) Jeżeli dla danego systemu prawdziwa jest właściwość wejściowa $Fu(u)$, to jaką postać ma właściwość wyjściowa $Fy(y)$?
3) Jaka postać faktów $Fu(u)$ gwarantuje spełnienie właściwości wyjściowej $Fy(y)$?

Odpowiedzi na te pytania poszukuje się przy założeniu, że wszystkie fakty opisujące obiekt są prawdziwe. W notacji metody logiczno-algebraicznej przedstawione pytania przyjmują postać implikacji [38]: $Fu(u) \Rightarrow Fy(y)$.

Ad. 1) Pierwsze pytanie prowadzi do sprawdzenia czy przy zadanej postaci faktów $Fu(u)$, $Fy(y)$ spełniona jest implikacja $Fu(u) \Rightarrow Fy(y)$:

Ad. 2) Drugie pytanie prowadzi do wyznaczenia takiej postaci właściwości $Fy(y)$ by poniższa reguła była spełniona:

$$\frac{Fu(u)\ ,\quad Fu(u) \Rightarrow Fy(y)}{Fy(y)}$$

Innymi słowy, poszukiwana jest postać faktów $Fy(y)$, dla której istnieć będą takie wartości zmiennych y, że implikacja $Fu(u) \Rightarrow Fy(y)$ i fakty $Fu(u)$ będą

prawdziwe.

Ad. 3) Trzecie pytanie prowadzi do wyznaczenia takiej postaci właściwości $Fu(u)$ by poniższa reguła była spełniona:

$$\frac{\neg Fy(y)\ ,\quad Fu(u) \Rightarrow Fy(y)}{\neg Fu(u)}$$

gdzie: $\neg\ Fu(u)$ – oznacza nieprawdziwość właściwości wejściowej, to znaczy wartość logiczna $Qu_i(u)$ co najmniej jednego faktu $Fu_i(u)$ wchodzącego w skład $F_u(u)$, jest równa zero.

Innymi słowy, poszukiwana jest postać faktów $Fu(u)$, dla której istnieją takie wartości zmiennych u, że implikacja $Fu(u) \Rightarrow Fy(y)$ i fakty $Fy(y)$ są spełnione.

Poszukiwanie odpowiedzi na zadane pytania, w terminologii metody logiczno-algebraicznej, wiąże się rozwiązaniem problemów: analizy i problemu decyzyjnego [38].

Problem analizy polega na wyznaczeniu, dla danego sytemu opisanego reprezentacją wiedzy KB i znanej właściwości wejściowej $Fu(u)$, najlepszej (tzn. implikującej każdą inną) właściwości $Fy(y)$, dla której spełniona jest implikacja $Fu(u) \Rightarrow Fy(y)$ [38]. Rozwiązanie tego problemu umożliwia udzielenie odpowiedzi na pytania 1 i 2.

Problem decyzyjny, sprowadza się do wyznaczenia dla bazy wiedzy KB, najlepszej (tzn. takiej, która jest implikowana przez każdą inną) właściwości wejściowej $Fu(u)$ zapewniającej spełnienie właściwości wyjściowej $Fy(\alpha y)$ [38]. Dla odnalezionej właściwości $Fu(u)$ spełniona jest implikacja: $Fu(u) \Rightarrow Fy(y)$. Rozwiązanie tego problemu umożliwia udzielenie odpowiedzi na pytanie 3.

3.1.2. Schemat wnioskowania

W oparciu o bazę wiedzy KB (3.3) możliwe jest poszukiwanie warunków wystarczających traktowanych jako właściwość wejściowa, tzn. właściwość gwarantująca określoną postać właściwości wyjściowej.

Postać właściwości wyjściowej $Fy(y)$, dla której poszukiwane są warunki wystarczające wynika zawsze z postaci pytania stawianego przez użytkownika. Dla przykładu: Dany jest system transportowy będący podsystemem systemu produkcyjnego. Użytkownik systemu poszukuje odpowiedzi na pytanie:

Czy istnieje harmonogram pracy wózków samojezdnych, w którym wszystkie wózki realizują swoje operacje w arbitralnie zadanym przez użytkownika czasie H? Jeżeli tak, to jaką postać ma ten harmonogram ?

Zgodnie ze strukturą z rysunku 2.4, system wspomagania decyzji przeznaczony do odpowiedzi na powyższe pytanie powinien spełniać określone warunki wystarczające, gwarantujące taką odpowiedź. Przyjmując, że warunki będą poszukiwane w kontekście takich parametrów jak stany początkowe systemu, rozważane pytanie przyjmuje postać:

Jaki stan początkowy gwarantuje, że wszystkie wózki samojezdne zrealizują swoje operacje w arbitralnie zadanym przez użytkownika czasie H?

Odpowiedź na to pytanie polega na znalezieniu takiej właściwości wejściowej $Fu(u)$ (gdzie u oznacza stan początkowy), która zagwarantuje, że przy spełnieniu wszystkich faktów (opisujących działanie systemu transportowego) spełniona będzie dodatkowa właściwość $Fy(y)$ (odpowiadająca sytuacji, w której wózki realizują swoje operacje w zadanym czasie H).

Powyższy przykład ilustruje, że poszukiwanie warunków wystarczających w postaci $Fu(u)$ implikuje konieczności rozwiązania odpowiedniego problemu decyzyjnego.

Rozwiązanie problemu decyzyjnego przy wykorzystaniu metody logiczno-algebraicznej polega na wyznaczeniu zbioru S_u, w oparciu o uprzednio wyznaczone zbiory S_{u1} i S_{u2}:

$$S_u = S_{u1} \backslash S_{u2} . \tag{3.4}$$

Zbiory S_{u1} i S_{u2} wyznaczane są przez rozwiązywanie względem zmiennych wejściowych u następujących układów równań [38]:

dla S_{u1}:

$$\begin{cases} Q(u,w,y) = \bar{1} , \\ Qy(y) = \bar{1} \end{cases} \tag{3.5}$$

dla S_{u2}:

$$\begin{cases} Q(u,w,y) = \bar{1} . \\ \neg Qy(y) = \bar{1} \end{cases} \tag{3.6}$$

gdzie: $Q(u,w,y) = \bar{1}$ – oznacza przyporządkowanie każdemu elementowi $Q_i(u,w,y)$ wartości logicznej 1,

$Qy(y) = \bar{1}$ – oznacza przyporządkowanie każdemu elementowi $Qy_i(y)$ wartości logicznej 1,

$\neg Qy(y) = \bar{1}$ – oznacza, że co najmniej jeden element $Qy_i(y)$ ma wartości logiczną 0.

W problemie decyzyjnym poszukiwana jest taka postać faktów $Fu(u)$, która spełnia zbiór faktów $F(u,w,y)$ oraz spełnia implikację $Fu(u) \Rightarrow Fy(y)$. Oznacza to, że poszukiwany jest zbiór S_u wartości u, który stanowi reprezentację liczbową poszukiwanego zbioru faktów $Fu(u)$. Efektem rozwiązania układu (3.5) są takie wartości zmiennej wejściowej u, dla których spełnione są wszystkie fakty $F(u,w,y)$ ($Q(u,w,y) = \bar{1}$) relacji R oraz spełnione są fakty wyjściowe $F_y(y)$ ($Qy(y) = \bar{1}$). Analogicznie, rozwiązaniem układu (3.6) jest zbiór wartości zmiennej wejściowej u, dla którego spełnione są wszystkie fakty $F(u,w,y)$ relacji R oraz nie spełnione są fakty wyjściowe $F_y(y)$ ($\neg Qy(y) = \bar{1}$).

Zbiór S_u otrzymany z różnicy zbiorów S_{u1} i S_{u2} jest zbiorem wartości zmiennych otrzymanych z relacji:

$$S_u = \{u : Qu(u) = \bar{1}\} , \tag{3.7}$$

gdzie: $Qy(y) = \bar{1}$ – oznacza przyporządkowanie każdemu elementowi $Qu_i(u)$ wartości logicznej 1, $Qu_i(u)$ jest wartością logiczną faktu $Fu_i(u)$.

Zbiór S_u stanowi zatem liczbową reprezentację poszukiwanej właściwości $F_u(u)$. Rozwiązania otrzymane w wyniku stosowania metody logiczno-algebrycznej odpowiadają rozwiązaniom otrzymywanym w wyniku stosowania klasycznych metod wnioskowania opartych np. na metodzie rezolucji. Poniżej przedstawiony przykład ilustruje porównanie obu metod wnioskowania.

Przykład 3.2. Porównanie metody rezolucji z wnioskowaniem metody logiczno-algebraicznej

Przykład ilustruje różnicę między wnioskowaniem wykorzystującym metodę rezolucji i wnioskowaniem metody logiczno-algebraicznej.
Dana jest reprezentacja wiedzy w postaci :

$$KB = <\alpha; F(\alpha)>,$$

gdzie ciąg formuł elementarnych ma postać:

$$\alpha = (\alpha p, \alpha r, \alpha s, \alpha t, \alpha q, \alpha z).$$

Formułom elementarnym α odpowiada ciąg wartości logicznych $a = (p, r, s, t, q, z)$.
Ciąg wartości logicznych faktów jest scharakteryzowany następująco:

$$F(\alpha) = (F_1(\alpha), F_2(\alpha), F_3(\alpha), F_4(\alpha)),$$

gdzie: $F_1(\alpha): \alpha z \Rightarrow \alpha p$,
$F_2(\alpha): \alpha p \wedge \alpha q \Rightarrow \alpha r$,
$F_4(\alpha): \alpha s \Rightarrow \alpha q$,
$F_3(\alpha): \alpha t \Rightarrow \alpha q$.

Dokonano arbitralnego podziału formuł na formuły wejściowe: $\alpha u = (\alpha z, \alpha s, \alpha t)$, $au = (z, s, t)$; pomocnicze: $\alpha w = (\alpha p, \alpha q)$, $aw = (p, q)$; wyjściowe: $\alpha y = \alpha r$, $ay = r$.

Należy odpowiedzieć na pytanie: *Jaka postać faktów Fu(αu) gwarantuje spełnienie faktu Fy(αy)?* gdzie: $Fy(\alpha y)$: αr.

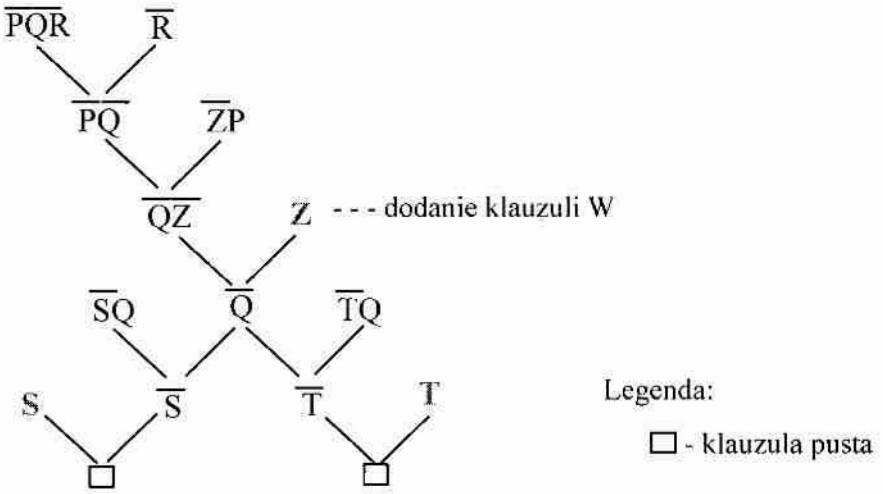
W pierwszej kolejności, do rozwiązania problemu, wykorzystano metodę rezolucji. Zadanie polega na znalezieniu takich wartości logicznych z, s, t, dla których spełniony jest fakt $Fy(\alpha y)$. W tym celu fakty $F(\alpha)$ sprowadza się do postaci klauzul:

$F_1(\alpha)$: $\neg\alpha z \vee \alpha p$,
$F_2(\alpha)$: $\neg\alpha p \vee (\neg\alpha q) \vee \alpha r$,
$F_4(\alpha)$: $\neg\alpha s \vee \alpha q$,
$F_5(\alpha)$: $\neg\alpha t \vee \alpha q$.

Fakty zapisano w postaci skrótowej w następującej postaci [11]:

$$\{\overline{Z}P, \overline{P}\,\overline{Q}R, \overline{S}Q, \overline{T}Q, \overline{R}\},$$

gdzie $\overline{ZP}$ należy czytać jako $\neg \alpha z \vee \alpha p$. Symbole P, R, S, T, Q, Z, odpowiadają kolejno formułom αp, αr, αs, αt, αq, αz. Dodatkowo, umieszczono wyrażenie $\overline{R}$ będące zaprzeczeniem faktu $Fy(\alpha y)$. Poszukiwanie odpowiedzi na postawione pytanie odbywa się poprzez budowanie drzewa rezolucji, aż do momentu uzyskania klauzuli pustej (oznaczanej przez symbol $\square$). Uzyskane drzewo rezolucji [11] przedstawione zostało na rysunku 3.3.

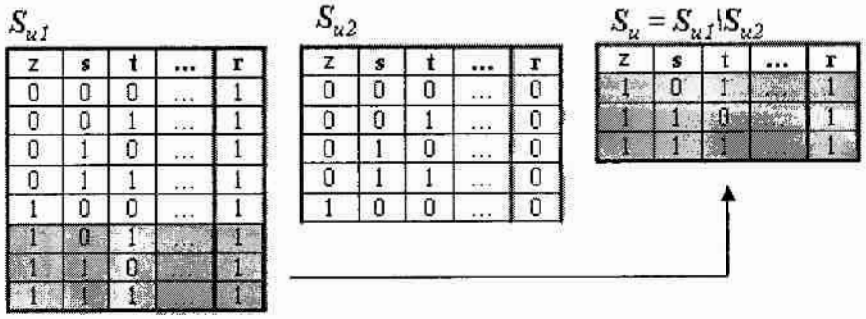


Rys. 3.3. Drzewo rezolucji

W trakcie tworzenia drzewa rezolucji w trzech miejscach drzewa należało uzupełnić zbiór klauzul o zdania w postaci ***Z***, ***S***, ***T***. Struktura otrzymanego drzewa oznacza, że jeśli zbiór faktów uzupełnimy o zdania $\alpha z \wedge \alpha s$ lub o zdanie $\alpha z \wedge \alpha t$ to otrzymana zostanie sprzeczność, czyli zdanie $\neg r$ nie jest spełnione, a zatem spełnione jest zdanie r. Poszukiwany fakt wejściowy $Fu(au)$ ma postać:

$$Fu(\alpha u): (\alpha z \wedge \alpha s) \vee (\alpha z \wedge \alpha t) .$$

Poszukiwanie postaci faktów $Fu(au)$ w przypadku metody logiczno-algebraicznej polega na wyznaczeniu zbiorów S_{u1} i S_{u2} spełniających układy równań (3.5), (3.6). W najprostszym przypadku wyznaczenie tych zbiorów polega na przeglądzie tablic prawdy. Tablice prawdy uzyskane dla poszukiwanych zbiorów zostały przedstawione na rysunku 3.4.



Rys. 3.4. Tabele prawdy określające wartości logiczne formuł z, s, t, dla zbiorów S_{u1}, S_{u2}, S_u

Na rysunku 3.4 kolorem oznaczone są te wiersze tablicy, które odpowiadają wartościom otrzymanym w wyniku różnicy S_{u1}/S_{u2}.
Zbiór S_u wyrażany w postaci trójek (z, s, t) ma zatem postać:

$$S_u = \{(1,0,1),(1,1,0),(1,1,1)\}.$$

Otrzymany zbiór S_u odpowiada następującej postaci faktu $Fu(\alpha u)$:

$$Fu(\alpha u): (\alpha z \wedge (\neg \alpha s) \wedge \alpha t) \vee (\alpha z \wedge \alpha s \wedge (\neg \alpha t)) \vee (\alpha z \wedge \alpha s \wedge \alpha t) = \mathbf{(\alpha z \wedge \alpha s) \vee (\alpha z \wedge \alpha t)}$$

Otrzymany wynik jest identyczny z wynikiem uzyskanym przy pomocy metody rezolucji.

■

Przykład 3.2 pokazuje, że stosowanie metody logiczno-algebraicznej prowadzi do tych samych wyników jak zastosowanie metody rezolucji. Różnica polega na sposobie otrzymania rozwiązania.

Wykorzystanie metod przeglądu opartych na analizie tablic prawdy w praktycznych przypadkach znacznie ogranicza stosowanie metody logiczno-algebricznej, z kolei metoda rezolucji jest obarczona koniecznością sprowadzania faktów do postaci klauzul.

Przedstawiony problemy decyzyjny wykorzystywany jest do poszukiwania warunków wystarczających rozumianych jako właściwość bazy wiedzy, której spełnienie gwarantuje spełnienie pożądanych przez decydenta właściwości (relacji). Wykorzystanie problemu decyzyjnego do poszukiwania warunków wystarczających zostało przedstawione na poniższym przykładzie.

Przykład 3.3. Zagadka Einsteina

Celem przykładu jest ilustracja reprezentacji wiedzy *KB* oraz warunków wystarczających gwarantujących istnienie zadanej właściwości, w tak zwanej, zagadce Einsteina [17]. Wiadomo, że pięciu obywateli mieszka w pięciu domach, w pięciu różnych kolorach, palą papierosy pięciu różnych marek, pija pięć różnych napoi, hodują pięć różnych zwierząt. Ponadto dane są informacje dodatkowe:

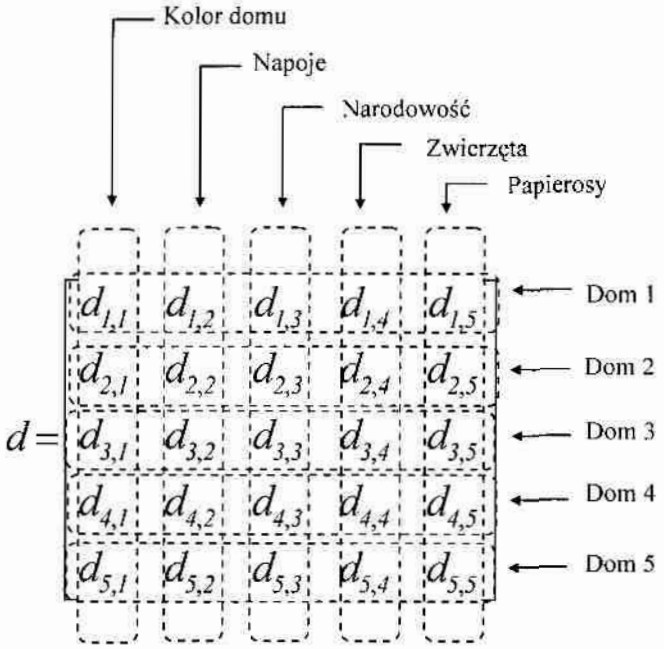
1. Norweg mieszka w pierwszym domu.
2. Anglik mieszka w czerwonym domu.
3. Zielony dom jest na lewo od białego.
4. Palacz Rothmansów mieszka obok hodowcy kotów.
5. Mieszkaniec żółtego domu pali Dunhile.
6. Niemiec pali Marlboro.
7. Palacz Pall-Malli hoduje ptaki.
8. Szwed hoduje psy.
9. Norweg mieszka obok niebieskiego domu.
10. Hodowca koni mieszka obok żółtego domu.
11. Palacz Philiip Moris pije piwo.

12. W zielonym domu pije się kawę.
13. Duńczyk pije herbatę.

W kontekście posiadanej wiedzy należy odpowiedzicć na pytanie:
Jakie napoje powinni pić mieszkańcy poszczególnych domów aby Niemiec mógł hodować rybki ?

Poszukiwane są więc takie warunki dotyczące napojów by spełniona była właściwość: "*Niemiec hoduje rybki*". W tym celu, w pierwszej kolejności, posiadaną wiedzę zgodnie z terminologią metody logiczno-algebraiczenej, należy sprowadzić do postaci formuł i faktów.

Wprowadzono następującą tablicę *d*:



gdzie: $d_{i,j} \in \{1,2, \ldots ,5\}$

Tablicę *d* należy interpretować następująco: wiersze macierzy określają kolejne domy, kolumny oznaczają kolejno: kolor domu, rodzaj napoi, narodowość mieszkańca, hodowane zwierzę, rodzaj papierosów. Zmienne $d_{i,j}$ przyjmują wartości ze zbioru {1, 2, … ,5}, interpretacja tej samej wartości dla każdej kolumny jest inna. Wartości zmiennych dla poszczególnych kolumn oznaczają:

Dla kolumny $d_{i,1}$ (kolor domu) :

1 – czerwony , 2 – biały, 3 – żółty, 4 – niebieski, 5 – zielony,

Dla kolumny $d_{i,2}$ (napoje) :

1 – herbata, 2 – piwo, 3 – kawa, 4 – mleko, 5 – woda,

Dla kolumny $d_{i,3}$ (narodowość) :

1 – Norweg, 2 – Anglik, 3 – Duńczyk, 4 – Szwed, 5 – Niemiec,

Dla kolumny $d_{i,4}$ (zwierzęta) :

1 – koty, 2 – ptaki, 3 – psy, 4 – konie, 5 – ryby,

Dla kolumny $d_{i,5}$ (papierosy) :

1 – Malboro, 2 – Rothmans, 3 – Pall-Mall, 4 – Philiip Moris, 5 – Dunhile,

W oparciu o te informacje sformułowana została reprezentacja wiedzy:

$KB = <U, W, Y; Re>$

gdzie: $U, W, Y = \{1, 2, \ldots, 5\}$,

$u = (d_{1,2}, d_{2,2}, d_{3,2}, d_{4,2}, d_{5,2})$ – zmienne wejściowe u określające rodzaj napoi, $d_{1,2}, d_{2,2}, d_{3,2}, d_{4,2}, d_{5,2} \in U$,

$y = (d_{1,3}, d_{2,3}, \ldots, d_{5,3}, d_{1,4}, d_{2,4}, \ldots, d_{5,4})$ – zmienne wyjściowe y określające narodowość mieszkańców i hodowane zwierzęta,

$d_{1,3}, d_{2,3}, \ldots, d_{5,3}, d_{1,4}, d_{2,4}, \ldots, d_{5,4} \in Y$,

$w = (d_{1,1}, d_{2,1}, \ldots, d_{5,1}, d_{1,5}, d_{2,5}, \ldots, d_{5,5})$ – zmienne pomocnicze,

$d_{1,1}, d_{2,1}, \ldots, d_{5,1}, d_{1,5}, d_{2,5}, \ldots, d_{5,5} \in W$,

$Re = \{(u,w,y): Q(u,w,y) = 1\}$ – relacja określająca związki między zmiennymi u, w, y. Zbiór faktów: $F(u,w,y) = F(d) = \{F_1(d), F_2(d), \ldots, F_{67}(d)\}$, gdzie fakty reprezentują następujące informacje:

1) **Norweg mieszka w pierwszym domu.** Informację w postaci tak sformułowanego zdania przedstawia się w postaci następującego faktu:

$F_1(d)$: $(d_{1,3} = 1)$.

2) **Anglik mieszka w czerwonym domu.**

Informacja ta interpretowana jest następująco: Jeśli w danym domu mieszka Anglik to ten dom jest czerwony. Prawdą jest również że: jeżeli dom jest czerwony to mieszka w nim Anglik. Ze względu na to, że brak jest informacji, o który dom dokładnie chodzi, zatem sytuacja ta odnosić się może do wszystkich pięciu. Prowadzi to do opisu w postaci następujących faktów:

- dla pierwszego domu:

  $F_2(d)$: $(d_{1,3} = 2) \Leftrightarrow (d_{1,1} = 1)$ to znaczy, że jeżeli w pierwszym domu mieszka Anglik to ten dom jest czerwony i odwrotnie, jeżeli dom jest czerwony to w tym domu mieszka Anglik,

- dla pozostałych domów:

  $F_3(d)$: $(d_{2,3} = 2) \Leftrightarrow (d_{2,1} = 1)$,
  $F_4(d)$: $(d_{3,3} = 2) \Leftrightarrow (d_{3,1} = 1)$,
  $F_5(d)$: $(d_{4,3} = 2) \Leftrightarrow (d_{4,1} = 1)$,
  $F_6(d)$: $(d_{5,3} = 2) \Leftrightarrow (d_{5,1} = 1)$.

3) **Zielony dom jest na lewo od białego**
Informacja ta prowadzi do następujących faktów:

$F_{7\text{-}11}(d)$: $(d_{i,1} = 5) \Leftrightarrow (d_{(i+1),1} = 2)$, gdzie $i = 1, 2, \ldots, 4$.

4) **Palacz Rothmansów mieszka obok hodowcy kotów.**
Informacja ta prowadzi do następujących faktów:

$F_{12}(d)$: $(d_{1,5} = 2) \Rightarrow (d_{2,4} = 1)$,
$F_{13}(d)$: $(d_{2,5} = 2) \Rightarrow (d_{1,4} = 1) \vee (d_{3,4} = 1)$,
$F_{14}(d)$: $(d_{3,5} = 2) \Rightarrow (d_{2,4} = 1) \vee (d_{4,4} = 1)$,
$F_{15}(d)$: $(d_{4,5} = 2) \Rightarrow (d_{3,4} = 1) \vee (d_{5,4} = 1)$,
$F_{16}(d)$: $(d_{5,5} = 2) \Rightarrow (d_{4,4} = 1)$.

5) **Mieszkaniec żółtego domu pali Dunhile.**
Informacja ta prowadzi do następujących faktów:

$F_{17\text{-}21}(d)$: $(d_{i,1} = 3) \Leftrightarrow (d_{i,5} = 5)$, gdzie $i = 1, 2, \ldots, 5$.

6) **Niemiec pali Marlboro.**
Informacja ta prowadzi do następujących faktów:

$F_{22\text{-}26}(d)$: $(d_{i,3} = 5) \Leftrightarrow (d_{i,5} = 1)$, gdzie $i = 1, 2, \ldots, 5$.

7) **Palacz Pall-Malli hoduje ptaki.**
Informacja ta prowadzi do następujących faktów:

$F_{27\text{-}31}(d)$: $(d_{i,5} = 3) \Leftrightarrow (d_{i,4} = 2)$, gdzie $i = 1, 2, \ldots, 5$.

8) **Szwed hoduje psy.**
Informacja ta prowadzi do następujących faktów:

$F_{32\text{-}37}(d)$: $(d_{i,4} = 3) \Leftrightarrow (d_{i,3} = 4)$ gdzie $i = 1, 2, \ldots, 5$.

9) **Norweg mieszka obok niebieskiego domu.**
Informacja ta prowadzi do następujących faktów:

$F_{38}(d)$: $(d_{1,3} = 1) \Rightarrow (d_{2,1} = 4)$,
$F_{39}(d)$: $(d_{2,3} = 1) \Rightarrow (d_{1,1} = 4) \vee (d_{3,1} = 4)$,
$F_{40}(d)$: $(d_{3,3} = 1) \Rightarrow (d_{2,1} = 4) \vee (d_{4,1} = 4)$,
$F_{41}(d)$: $(d_{4,3} = 1) \Rightarrow (d_{3,1} = 4) \vee (d_{5,1} = 4)$,
$F_{42}(d)$: $(d_{5,3} = 1) \Rightarrow (d_{4,1} = 4)$.

10) **Hodowca koni mieszka obok żółtego domu.**
Informacja ta prowadzi do następujących faktów:

$F_{43}(d)$: $(d_{1,4} = 4) \Rightarrow (d_{2,1} = 3)$,
$F_{44}(d)$: $(d_{2,4} = 4) \Rightarrow (d_{1,1} = 3) \vee (d_{3,1} = 3)$,
$F_{45}(d)$: $(d_{3,4} = 4) \Rightarrow (d_{2,1} = 3) \vee (d_{4,1} = 3)$,
$F_{46}(d)$: $(d_{4,4} = 4) \Rightarrow (d_{3,1} = 3) \vee (d_{5,1} = 3)$,
$F_{47}(d)$: $(d_{5,4} = 4) \Rightarrow (d_{4,1} = 3)$.

11) **Palacz Philiip Moris pije piwo.**
Informacja ta prowadzi do następujących faktów:

$F_{48\text{-}51}(d)$: $(d_{i,5} = 4) \Leftrightarrow (d_{i,2} = 2)$, gdzie $i = 1,2,..., 5$.

12) **W zielonym domu pije się kawę.**
Informacja ta prowadzi do następujących faktów:

$F_{57\text{-}61}(d)$:$(d_{i,1} = 5) \Leftrightarrow (d_{i,2} = 3)$, gdzie $i = 1,2,..., 5$.

13) **Duńczyk pije herbatę.**
Informacja ta prowadzi do następujących faktów:

$F_{62\text{-}67}(d)$: $(d_{i,3} = 3) \Leftrightarrow (d_{i,2} = 1)$, gdzie $i = 1,2,..., 5$.

Przedstawione dane tworzą bazę wiedzy, której strukturę zilustrowano na rysunku 3.5. Odpowiada ona strukturze z rysunku 3.1. W tym kontekście, problem poszukiwania warunków wystarczających polega na znalezieniu takiej postaci części wejściowej bazy wiedzy (którą musi zadać użytkownik systemu wspomagania decyzji), która zagwarantuje, że część wyjściowa bazy wiedzy będzie się charakteryzowała oczekiwaną właściwością („*Niemiec hoduje rybki*").

Przyjęto, że zbiór faktów wyjściowych określający właściwość wyjściową: „*Niemiec hoduje rybki*" ma postać:

$$Fy(y) : (d_{1,3} = 5) \wedge (d_{1,4} = 5) \vee (d_{2,3} = 5) \wedge (d_{2,4} = 5) \vee \ldots \vee (d_{5,3} = 5) \wedge (d_{5,4} = 5).$$

Dla takiej postaci faktu wyjściowego poszukiwana jest postać faktu wejściowego $Fu(u)$. W tym celu należało rozwiązać problem decyzyjny. Postępując analogicznie jak w przypadku przykładu 3.2 wyznaczone zostały tablice prawdy określające wartości logiczne zmiennych d oraz faktu $Fy(y)$. Z tablic tych wyznaczone zostały postacie zbiorów S_{u1} i S_{u2} zawierające takie wartości zmiennych $d_{1,2}$, $d_{2,2}$, $d_{3,2}$, $d_{4,2}$, $d_{5,2}$, dla których fakt $Fy(y)$ jest odpowiednio prawdziwy (dla S_{u1}) lub fałszywy (dla S_{u2}). Ze zbiorów S_{u1}, S_{u2}, wyznaczony został zbiór wartości wejściowych S_u (przedstawiany jako zbiór piątek $(d_{1,2}, d_{2,2}, d_{3,2}, d_{4,2}, d_{5,2})$):

$$S_u = \{ (4,1,5,3,2), (5,1,4,3,2) \}.$$

Otrzymany zbiór jest interpretowany następująco:
Jeżeli mieszkańcy kolejnych domów piją:

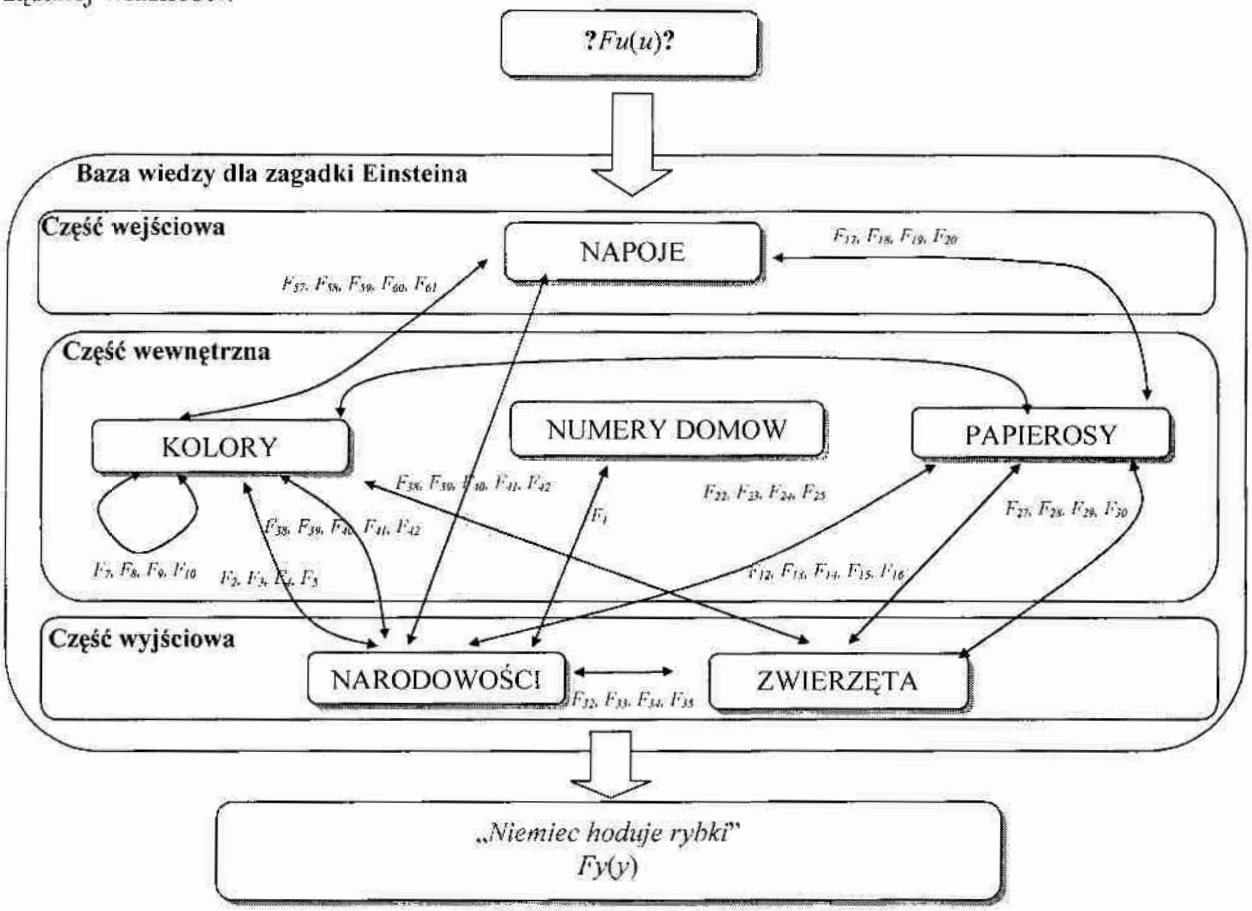
mleko (dom 1), herbatę (dom 2), wodę (dom 3), kawę (dom 4), piwo (dom5),

lub

wodę (dom 1), herbatę (dom 2), mleko (dom 3), kawę (dom 4), piwo (dom5),

to Niemiec hoduje rybki.

Powyższe rozwiązanie stanowi więc warunki wystarczające gwarantujące spełnienie żądanej własności.



Rys. 3.5. Struktura wzajemnych związków między poszczególnymi zmiennymi w bazie wiedzy zagadki Einsteina

■

Uzyskana w przykładzie 3.3 właściwość stanowi warunek wystarczający, spełnienie którego gwarantuje istnienie w bazie wiedzy określonych, zadanych właściwości.

Poszukiwanie tego typu właściwości jest istotne z punktu widzenia budowy interakcyjnych systemów wspomagania decyzji. Przedstawiony mechanizm oparty na metodzie logiczno-algebraicznej, może być wykorzystywany do poszukiwania warunków wystarczających gwarantujących istnienie odpowiedzi na zadane przez użytkownika pytanie. Wyznaczanie warunków wystarczających w oparciu proponowaną metodykę możliwe jest w przypadku, gdy posiadana wiedza o obiekcie, (przedsiębiorstwo, sieć transportowa, itp.) w kontekście którego prowadzone jest wspomaganie decyzji, może być reprezentowana w postaci reprezentacji wiedzy *KB* (3.3) odpowiadającej strukturze bazy wiedzy z rysunku 3.1. W przedstawionym podejściu trudnym zagadnieniem jest rozwiązanie samego problemu

decyzyjnego (którego wynikiem są poszukiwane warunki wystarczające). Okazuje się, że dostępne metody rozwiązywania układów równań (3.5), (3.6) charakteryzują się złożonością wykładniczą [38]. Stąd też powstaje potrzeba poszukiwania efektywnych strategii ich rozwiązywania.

3.1.3. Schematy faktów

Przy rozwiązywaniu praktycznych problemów istotne jest to by opis obiektu w postaci zmiennych decyzyjnych i relacji móc wyrazić w postaci zbioru faktów $F(u,w,y)$. Okazuje się, że w wielu przypadkach struktura obiektu charakteryzuje się pewnymi regularnościami pozwalającymi zaliczyć ją do określonej klasy. Regularności te można uchwycić pewnymi strukturami, tzw. **schematami faktów** F_P. Regularności mogą dotyczyć na przykład zachowania obiektu i wyróżniać pewne powtarzające się w czasie (i/lub przestrzeni) elementy. Schematy faktów F_P mogą być budowane automatycznie, tworząc zbiory sparametryzowanych zdań logicznych:

$$F_P(u,w,y,pr) = \{F_{P,1}(u, w, y, pr), F_{P,2}(u, w, y, pr), \ldots, F_{P,K(pr)}(u, w, y, pr)\},$$

gdzie: $u = (u_1, u_2, \ldots, u_{ku})$, $w = (w_1, w_2, \ldots, w_{kw})$, $y = (y_1, y_2, \ldots, y_{ky})$ – zmiennne wejściowe, wewnętrzne i wyjściowe,

$pr = (pr_1, pr_2, \ldots, pr_{kr})$ – ciąg parametrów,

$K(pr)$ – funkcja, której wartość określa liczbę faktów wchodzących w skład zbioru $F_P(u, w, y, pr)$, w zależności od wartości parametów pr.

$F_{P,I}(u, w, y, pr)$ – sparametryzowany fakt, zdanie logiczne opisujące relacje między zmiennymi u, w, y, w zależności od wartości parametrów pr.

Parametry stanowią te zmienne decyzyjne obiektu, które charakteryzują występujące w nim regularności. Wartości parametrów wpływają na wartości zmiennych decyzyjnych, stosowane operatory logiczne, postać funkcji zdaniowych, a także na liczbę faktów opisujących obiekt. Ogólna postać sparametryzowanego faktu $F_{P,I}(u, w, y, pr)$ jest następująca:

$$F_{P,I}(u, w, y, pr) = op_1(pr)\{\ op_2(pr)\{\ \ldots, op_i(pr)\{\ \upsilon_o(u,w,y,pr)\ ,\ \upsilon_p(u,w,y,pr)\ \}\ \ldots\ \}\},\quad (3.8)$$
$$op_k(pr)\{\ \ldots, op_r(pr)\{\ \upsilon_j(u,w,y,pr)\ ,\ \upsilon_s(u,w,y,pr)\ \}\ \ldots\ \}\ \},$$

gdzie: $op_i(pr)\{b, c\}$ – funkcja wiążąca formuły b, c, operatorem logicznym ($\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$), którego rodzaj jest zależny od wartości parametrów pr. W szczególności dla operatora negacji ($\neg$) funkcja ma postać: $op_i(pr)\{b\}$, funkcja określa występowanie operatora negacji w formule b,

$\upsilon_i(u,w,y,pr)$ – funkcja zdaniowa określająca postać formuły elementarnej α_i w zależności od wartości parametrów pr.

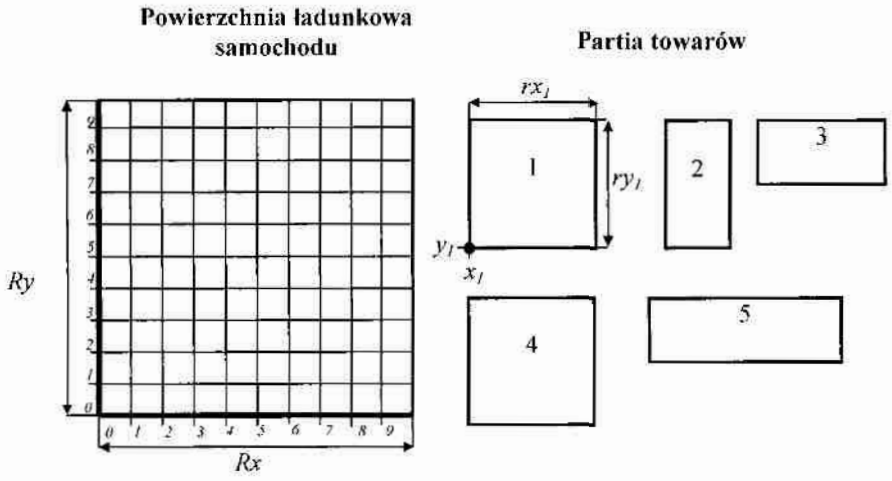
Schematy faktów odgrywają istotną rolę przy budowie reprezentacji wiedzy. Dostrzeżone przez projektanta systemu regularności obiektu pozwalają na opis wiedzy w postaci sparametryzowanych zdań logicznych. Taka postać faktów umożliwia automatyczne generowanie reprezentacji wiedzy KB w zakresie klasy obiektów

charakteryzujących się wspólnymi regularnościami. Raz zrealizowany schemat faktów może być wykorzystywany wielokrotnie dla różnych wariantów zachowania się obiektu. W zależności od wartości parametrów *pr* schemat faktów wykorzystywany jest w różnych problemach charakteryzowanych tym samym zbiorem ogólnych zasad. Schematom faktów stawiany jest warunek by w pełni odwzorowywały naturę problemu. Poniżej przedstawiono przykład specyfikacji schematu faktów F_P dla klasycznego problemu magazynowania.

Przykład 3.4. Problem składowania

Przykład ma na celu ilustrację specyfikacji problemu składowania. Dany jest samochód o określonej powierzchni ładunkowej przewożący towary produkowane w pewnym przedsiębiorstwie. Towary są przewożone w kontenerach w kształcie prostopadłościanów. Ze względu na rodzaj przewożonego towaru kontenery nie mogą być układane jedne na drugim. Należy odpowiedzieć na pytanie: *Czy zadana partia towarów może zostać przewieziona w jednym transporcie? Jeżeli tak, to w jaki sposób ułożyć kontenery w samochodzie?*

Na rysunku 3.6 przedstawiono przestrzeń ładunkową oraz kontenery (widok z góry).



Rys. 3.6. Powierzchnia ładunkowa samochodu, partia przewożonych towarów

Rozmiar powierzchni ładunkowej (kształt prostokątny) jest opisany przez wielkości *Rx* i *Ry*, określające kolejno szerokość i długość powierzchni. Poszczególne kontenery opisywane są przez wielkości y_i, x_i, będące współrzędnymi położenia lewego dolnego rogu *i*-tego kontenera, oraz przez wielkości rx_i, ry_i, określające rozmiar *i*-tego kontenera (długość i szerokość). Ponadto każdemu kontenerowi przypisywany jest binarny parametr orientacji o_i, określający orientację ułożenia kontenera (wzdłuż krawędzi pionowej, lub wzdłuż krawędzi poziomej powierzchni ładunkowej). Należy podkreślić, że wszystkie wielkości są zmiennymi dyskretnymi: $Rx, Ry, y_i, x_i, rx_i, ry_i \in N$.

W rozważanym problemie przyjęto następujące wartości parametrów (wyrażane w umownych jednostkach długości ***ujd***):

$Rx = 10, Ry = 10,$

$y_1 = 4, x_1 = 4,\ y_2 = 4, x_2 = 2,\ y_3 = 2, x_3 = 4,\ y_4 = 4, x_4 = 4,\ y_5 = 2, x_5 = 6.$

Wymagane jest by przy układaniu pięciu kontenerów zawsze spełnione były następujące ogólne zasady:

1) Kontenery nie mogą zachodzić na siebie nawzajem - współrzędne położenia kontenerów powinny być tak określone by prostokąty reprezentujące poszczególne kontenery nie miały części wspólnych.
2) Kontenery nie mogą przekraczać granic powierzchni ładunkowej - współrzędne położenia kontenerów powinny być tak określone by prostokąty reprezentujące poszczególne kontenery nie przekraczały krawędzi prostokąta reprezentującego przestrzeń ładunkową.

Tak zdefiniowany problem należy wyrazić w terminologii metody logiczno-algebraicznej. W tym celu przedstawione powyżej ogólne zasady zapisano w postaci sparametryzowanych zdań logicznych. Sformułowane fakty powinny gwarantować, że dla każdego kontenera bez względu na jego orientację i położenie w przestrzeni przedstawione zasady będą zawsze spełnione. Przyjęty zbiór faktów ma postać:

$$F_P(x, y, rx, ry, Rx, Ry, L) = F_{PA1}(x, y, rx, ry, Rx, Ry, L) \cup F_{PA2}(x, y, rx, ry, Rx, Ry, L).$$

Ad. 1) Fakty odpowiadające zasadzie 1:

$$F_{PA1}(x, y, rx, ry, Rx, Ry, L) = \{F_{A1,1}(x, y, rx, ry, Rx, Ry, L), F_{A1,2}(x, y, rx, ry, Rx, Ry, L), \ldots, F_{A1,at}(x, y, rx, ry, Rx, Ry, L), \ldots, F_{A1,at+L}(x, y, rx, ry, Rx, Ry, L)\},$$

gdzie: $at = \binom{L}{2}$, L = 5 – liczba kontenerów,

$F_{A1,r}(x, y, rx, ry, Rx, Ry, L)$: $(x_i \geq x_j + rx'_j) \vee (x_j \geq x_i + rx'_i) \vee (y_i \geq y_j + ry'_j) \vee (y_j \geq y_i + ry'_i)$,

$F_{A1,at+l}(x, y, rx, ry, Rx, Ry, L)$: $[(o_k = 0) \Rightarrow (rx'_k = rx_k) \wedge (rx'_k = rx_k)] \vee [(o_k = 1) \Rightarrow (rx'_k = ry_k) \wedge (ry'_k = rx_k)]$,

gdzie: $i = 1,2, \ldots, L\text{-}1$; $j = i+1, \ldots, L$; $k = 1,2, \ldots, L$,

$x = (x_1, x_2, \ldots, x_L)$, $y = (y_1, y_2, \ldots, y_L)$, $ry = (ry_1, ry_2, \ldots, ry_L)$, $rx = (rx_1, rx_2, \ldots, rx_L)$ – ciągi określające wartości współrzędnych kontenerów oraz ich rozmiary,

Rx, Ry – rozmiar przestrzeni ładunkowej,

o_k – parametr orientacji k-tego kontenera,

$$o_k = \begin{cases} 0 \text{ - orientacja wzdłuż krawędzi pionowej} \\ 1 \text{ - orientacja wzdłuż krawędzi poziomej} \end{cases},$$

rx'_k, ry'_k – wielkości pomocnicze.

Ad. 2) Ograniczenia odpowiadające zasadzie 2:

$$F_{PA2}(x, y, rx, ry, Rx, Ry, L) = \{F_{A2,1}(x, y, rx, ry, Rx, Ry, L), \ldots, F_{A1,L}(x, y, rx, ry, Rx, Ry, L)\},$$

$F_{A2,r}(x, y, rx, ry, Rx, Ry, L)$: $(x_i + rx'_i \leq Rx) \wedge (y_i + ry'_i \leq Ry)$,

gdzie: $i = 1,2, \ldots, L$,

rx'_i, ry'_i – wielkości pomocnicze.

W przyjętym zbiorze faktów zmienne *rx*, *ry*, *Rx*, *Ry*, *L*, są parametrami, *pr* = (*rx*, *ry*, *Rx*, *Ry*, *L*), których wartości wpływają na postać faktów oraz ich liczbę.

■

Przedstawiony w przykładzie 3.4 schemat faktów F_P z powodzeniem może zostać użyty do specyfikacji wiedzy dla innych problemów magazynowania. Wystarczy zmienić na przykład wartości takich wielkości jak *L*, *Rx*, *Ry*, by móc wykorzystać zdefiniowane fakty w problemach z inną przestrzenią ładunkową lub inną liczbą kontenerów. Jest to podstawowa zaleta wykorzystywania schematów faktów. W zależności od wartości parametrów w sposób automatyczny możliwe jest wyznaczenie zbioru faktów opisujących wiedzę o obiekcie i otrzymywać w ten sposób reprezentację wiedzy *KB* odpowiadającą temu obiektowi.

Przy projektowaniu systemów wspomagania decyzji istotne jest zatem posiadanie reprezentacji wiedzy wyrażonej w postaci schematów faktów. Stosowanie do opisu wiedzy schematów faktów umożliwi elastyczne dostosowywanie (w obrębie klasy obiektów charakteryzowanych przez parametry *pr*) postaci reprezentacji wiedzy *KB* do wymagań stawianych przez użytkownika systemu. W ten sposób, w zależności od wartości parametrów *pr*, otrzymywane są różne postacie reprezentacji wiedzy *KB*, w kontekście których możliwe jest prowadzenie poszukiwania warunków gwarantujących istnienie odpowiedzi na zadane przez użytkownika pytania.

3.2. Programowanie z ograniczeniami

Programowanie z ograniczeniami (*CP*), to zbiór technik służących do rozwiązywania problemów kombinatorycznych [10], jak np. problemy marszrutowania, magazynowania, porcjowania, harmonogramowania pracy pojazdów, itp. Z problemami tymi mamy zwykle do czynienia przy planowaniu produkcji w przedsiębiorstwach produkcyjnych. Są to problemy o charakterze NP-trudnym.

Deklaratywny charakter technik programowania z ograniczeniami oraz możliwość intensyfikacji problemów różnej natury, sprawiają, że są one powszechnie stosowane w systemach wspomagania decyzji.

Rozwój technik programowania z ograniczeniami doprowadził do powstania wielu systemów środowisk programowania z ograniczeniami. W większości przypadków udostępniane środki modelowania ograniczeń oraz implementacje mechanizmów rozwiązywania problemów oparto o środki dostarczane przez popularne języki programowania, np.: **Eclipse** [56], **Chip** [78], **Ilog** [69], **Oz Mozart** [75]. Dostępne języki programowania stanowią podstawę wielu komercyjnych systemów znajdujących zastosowania w transporcie, sporządzaniu wykazów i planowaniu zadań [37], [39], [71], [85]. Są stosowane do wspomagania procesów decyzyjnych w przedsiębiorstwach produkcyjnych, placówkach administracji państwowej, itp. [82], [101], [5] .

Pierwsza przemysłowa aplikacja, wykorzystująca język programowania w logice ograniczeń **Chip** (ang. Constraint Handling in Prolog) została opracowana dla portu

kontenerowego **HIT** w Hong Kongu [67], [81]. Celem jej było rozmieszczenie kontenerów w statkach kontenerowych w taki sposób, aby możliwie największa liczba kontenerów mogła być składowana na danej powierzchni.

Kolejnym systemem jest **GYMNASTE** [33], który służy do planowania dyżurów personelu w wielu szpitalach we Francji [81].

Techniki *CP* są stosowane w rozwiązywaniu problemów zarządzania personelem w przedsiębiorstwie telefonii brytyjskiej British Telecom [45], [81]. Telefonia włoska (ang. Telecom Italia), w celu planowania wszystkich zadań (około 100 000 zadań wykonywanych przez 20 000 techników) [64], wykorzystuje system kontrolujący zarządzanie personelem. System posiada moduł planowania zwany *ARCO*, który służy m.in. do harmonogramowania pracy techników oraz wykorzystuje procedury propagacji ograniczeń.

System **LOCARIM** został opracowany przez firmę **COSYTEC** dla firmy France Telecom, w celu rozwiązywania problemów związanych z instalacją sieci telekomunikacyjnej zespołu budynków [71], [81].

System **PLANETS** opracowany został przez Uniwersytet Kataloński w Barcelonie dla firmy zarządzającej sieciami elektroenergetycznymi w Hiszpanii. System ten jest narzędziem (bazującym na technikach *CP*) do modernizacji sieci elektroenergetycznych i pozwala na zarządzanie rozpływem mocy w tych sieciach [71], [81]. Poprzez oddzielenie elementów sieci (będących pod napięciem) od układów sterowania umożliwia on modernizację i zarządzanie sieciami bez przerywania dopływu energii elektrycznej do odbiorców (bez przerywania świadczonych klientom usług).

System **FORWARDC** jest systemem wspierania decyzji opartym na języku **Chip**, który wykorzystywany jest w trzech rafineriach ropy naftowej w Europie [40], [81]. Ma on na celu kontrolę wszystkich problemów planowania w procesie produkcji paliw, począwszy od przyjęcia surowca, poprzez przetwarzanie, aż do końcowego sporządzenia mieszanki i dostawy końcowej.

System **ATS4** opracowany przez firmę AMEplus i Politechnikę Śląską w Gliwicach wydział Automatyki, Elektroniki i Informatyki [101]. System ten jest przeznaczony do planowania zajęć dla dużych organizacji, jest również w pełni dostosowany do indywidualnych potrzeb użytkowników. Obecnie jest on stosowany do planowania zajęć na wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej.

Przedstawione przykłady oraz wiele innych wskazują na użyteczność i funkcjonalność narzędzi wykorzystujących techniki *CP* do rozwiązywania problemów w różnych obszarach działalności człowieka. Stanowią przykłady systemów wspomagania decyzji umożliwiających pracę z użytkownikiem w trybie interakcyjnym.

3.2.1. Problem spełniania ograniczeń

Istotą metod programowania z ograniczeniami jest rozwiązywanie problemów zdefiniowanych jako problemy spełniania ograniczeń (*PSO*). Problem *PSO* definiuje się następująco [29]:

Dany jest skończony zbiór zmiennych dyskretnych $V = \{V_1, V_2, \ldots, V_n\}$, rodzina dziedzin zmiennych decyzyjnych $D = \{D_i \mid D_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,j}, \ldots, d_{i,m}\}, i = 1..n\}$ określających wartości jakie mogą przyjmować elementy zbioru V oraz skończony zbiór ograniczeń $C = \{C_i \mid i = 1..Lc\}$ wiążących wartości zmiennych decyzyjnych. Poszukiwane jest rozwiązanie bądź to dopuszczalne, tzn. rozwiązanie odpowiadające tym wartościom, które spełniają wszystkie ograniczenia (w ogólności jest to zbiór rozwiązań), bądź też rozwiązanie optymalne ekstremalizujące funkcję celu określoną na wybranym podzbiorze zmiennych decyzyjnych. Przyjęto następującą notację problemu spełniania ograniczeń:

$$PSO = ((V, D), C), \tag{3.9}$$

gdzie: $C_i \in C$ jest pewnym predykatem $P_i[V_k, V_l, \ldots, V_h]$ zdefiniowanym na podzbiorze zbioru V [29].

Do rozwiązywania *PSO* wykorzystywane są środowiska programowania z ograniczeniami implementujące podstawowe mechanizmy propagacji ograniczeń i dystrybucji zmiennych. Pierwsze narzędzia, które służyły do programowania z ograniczeniami, to języki zawierające **Prolog**, między innymi **Chip**, **Eclipse**, **Siestus**. Ze względu na pewne ograniczenia, które wprowadza składania prologowa rozwinęły się dwie gałęzie języków programowania – pierwsza z nich to biblioteki **C/C++** (np. **Ilog Solver**), a druga to języki współbieżne (np. **Oz Mozart**).

Poszukiwanie rozwiązań polega na realizacji propagacji ograniczeń i dystrybucji zmiennych naprzemiennie. Propagacja ograniczeń jest jedną z najsilniejszych zalet programowania z ograniczeniami. Zasadą działania propagacji jest usuwanie wartości z dziedzin zmiennych D, dla których nie spełnione będzie co najmniej jedno ograniczenie ze zboru C. Efektywność procesu propagacji jest zależna od rodzaju używanego narzędzia. Propagacja ograniczeń może być częściowa, to znaczy usuwana jest tylko część wartości nie spełniających ograniczeń C, na przykład z dziedzin zmiennych D usuwane są tylko skrajne elementy. Usunięcie z dziedzin zmiennych D wartości nie spełniających ograniczeń C rzadko prowadzi do wyznaczenia rozwiązania. W sytuacji gdy propagacja ograniczeń jest niewystarczająca, to znaczy w wyniku procesu usuwania elementów dziedzin istnieją nadal wicloelementowe dziedziny D_i: $\exists D_i \in D : \|D_i\| > 1$, to w kolejnym kroku realizowany jest proces dystrybucji zmiennych.

Dystrybucja zmiennych polega na podzieleniu problemu *PSO* na dwa uzupełniające się podproblemy, które różnią się od podstawowego zbiorami ograniczeń [94]. W pierwszym podproblemie zbiór ograniczeń C jest uzupełniany o arbitralnie zadane ograniczenie C_D. Z kolei w przypadku drugiego podproblemu zbiór ograniczeń jest uzupełniany o ograniczenie $\neg C_D$. Ograniczenie C_D jest heurystycznie wybranym dodatkowym ograniczeniem

dystrybucyjnym. Najczęściej C_D jest ograniczeniem przypisującym określonej zmienne x_i wartość z zawężonej dziedziny D_i. Podproblemy uzupełnione o nowe ograniczenia są dalej podawane ponownemu procesowi propagacji ograniczeń. Procesy propagacji ograniczeń i dystrybucji zmiennych są realizowane cyklicznie, aż do momentu gdy zostanie znalezione rozwiązanie dopuszczalne (dziedziny D_i wszystkich zmiennych będą zbiorami jednoelementowymi: $\forall D_i \in D : \|D_i\| = 1$), bądź osiągnięty zostanie stan określający brak rozwiązania (istnieje dziedzina D_i będąca zbiorem pustym $\exists D_i \in D : \|D_i\| = 0$).

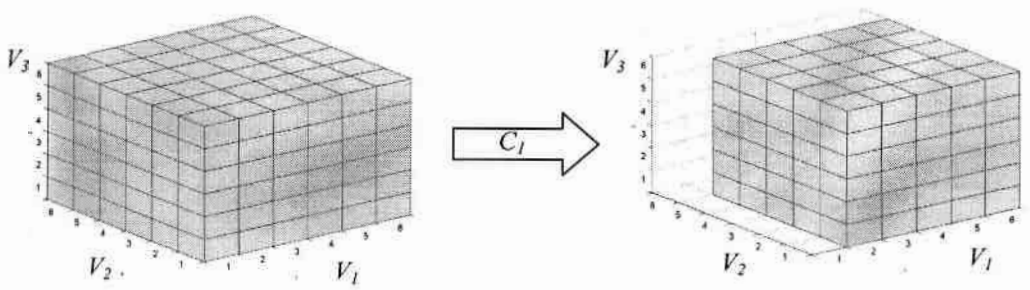
Przykład 3.5. Idea działania propagacji ograniczeń

Przykład ma na celu ilustrację mechanizmu propagacji ograniczeń. Dany jest problem $PSO = ((V, D), C)$, gdzie zbiór zmiennych decyzyjnych ma postać: $V = \{V_1, V_2, V_3\}$, zbiór dziedzin D określający dopuszczalne wartości zmiennych V_1, V_2, V_3, ma postać: $D = \{D_1, D_2, D_3\}$, $D_1 = D_2 = D_3 = \{1, 2, 3, 4, 5, 6\}$. Relacje opisujące związek między zmiennymi zadane są zbiorem: $C = \{C_1, C_2, C_3\}$, gdzie: C_1: $V_1 \geq V_2 + 1$, C_2: $V_2 \geq V_3 + 2$, C_3: $V_3 < V_1 - V_2$. Proces wstępnej (tzn. pierwszej, rozpoczynającej proces obliczeń) propagacji ograniczeń został przedstawiony na rysunku 3.7.

Rysunek 3.7 ilustruje kolejne etapy ograniczania dziedzin zmiennych V_1, V_2, V_3 w procesie propagacji. W pierwszym kroku propagacji (rysunek 3.7a) w wyniku ograniczenia C_1 dziedziny zmiennych zostają zawężone do postaci: $D_1 = \{2, 3, 4, 5, 6\}$, $D_2 = \{1, 2, 3, 4, 5\}$, $D_3 = \{1, 2, 3, 4, 5, 6\}$. Dziedzina zmiennej z nie została ograniczona. Wynika to z faktu, że zmienna z nie występuje w ograniczeniu C_1. Postępując analogicznie w kolejnych krokach przestrzeń potencjalnych rozwiązań została zwężona poprzez ograniczenia C_2 i C_3. Należy zauważyć, że w kolejnych etapach propagacji zawsze usuwane są tylko skrajne wartości zbiorów D_1, D_2, D_3. Postępowanie takie prowadzi do sytuacji, w której przestrzeń potencjalnych rozwiązań jest zawsze przestrzenią spójną. Rezultatem końcowym procesu propagacji są zbiory $D_1 = \{5, 6\}$, $D_2 = \{3, 4\}$, $D_3 = \{1, 2\}$ (rysunek 3.7c). Zatem przestrzeń z 216 elementów została zredukowana do zbioru zawierającego 8 elementów.

Proces propagacji okazał się niewystarczający do wyznaczania rozwiązania (dziedziny zmiennych są nadal wieloelementowe), np. rozwiązanie 5, 3, 2 nie spełnia ograniczeń C. Konieczny więc okazał się proces dystrybucji. Wynikiem etapu dystrybucji zmiennych są trzy rozwiązania dopuszczalne, które zilustrowano na rysunku 3.8.

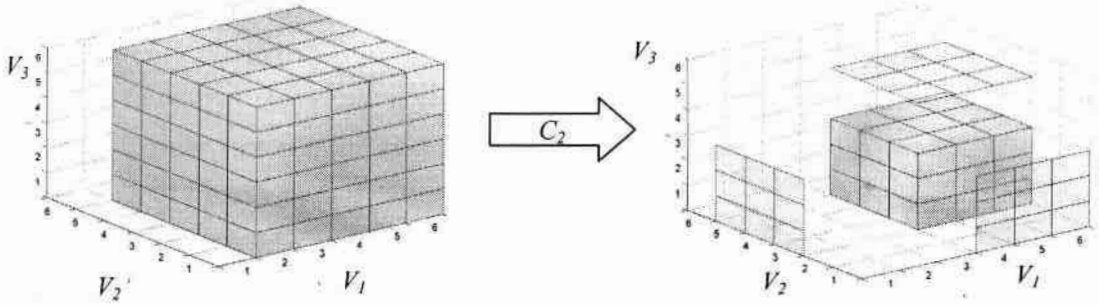
a)



$D_1 = \{1,2,3,4,5,6\}$
$D_2 = \{1,2,3,4,5,6\}$
$D_3 = \{1,2,3,4,5,6\}$

$D_1 = \{2,3,4,5,6\}$
$D_2 = \{1,2,3,4,5\}$
$D_3 = \{1,2,3,4,5,6\}$

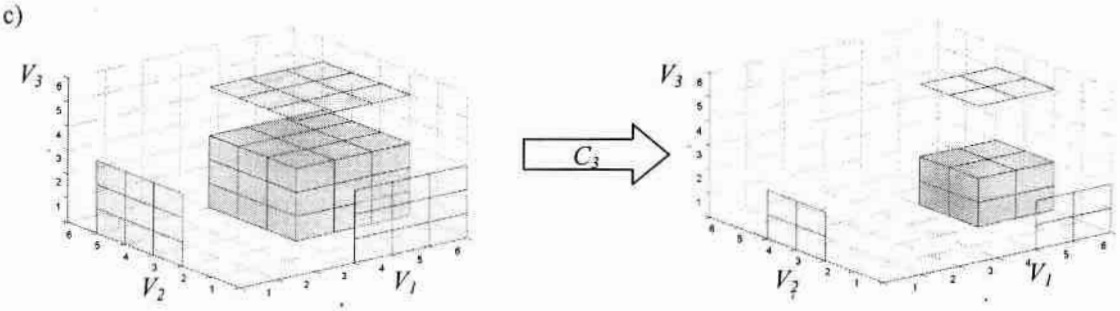
b)



$D_1 = \{2,3,4,5,6\}$
$D_2 = \{1,2,3,4,5\}$
$D_3 = \{1,2,3,4,5,6\}$

$D_1 = \{4,5,6\}$
$D_2 = \{3,4,5\}$
$D_3 = \{1,2,3\}$

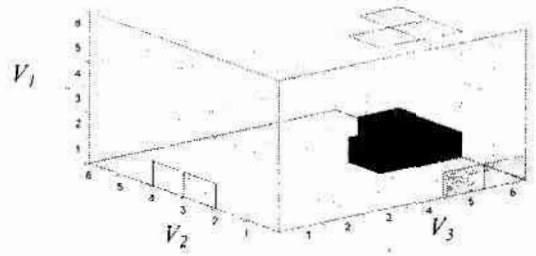
c)



$D_1 = \{4,5,6\}$
$D_2 = \{3,4,5\}$
$D_3 = \{1,2,3\}$

$D_1 = \{5,6\}$
$D_2 = \{3,4\}$
$D_3 = \{1,2\}$

Rys. 3.7. Ograniczanie przestrzeni rozwiązań w procesie propagacji ograniczeń, a) przestrzeń uzyskana w wyniku usunięcia wartości niespełniających ograniczenia C_1, b) przestrzeń po usunięciu wartości niespełniających ograniczenia C_2, c) przestrzeń po usunięciu wartości niespełniających ograniczenia C_3



V_1, V_2, V_3 V_1, V_2, V_3 V_1, V_2, V_3
(5 , 3 , 1), (6, 3, 1), (6, 4, 1),

Rys. 3.8. Rozwiązania dopuszczalne

Wszystkie otrzymane rozwiązania spełniają zadany zbiór ograniczeń C.

■

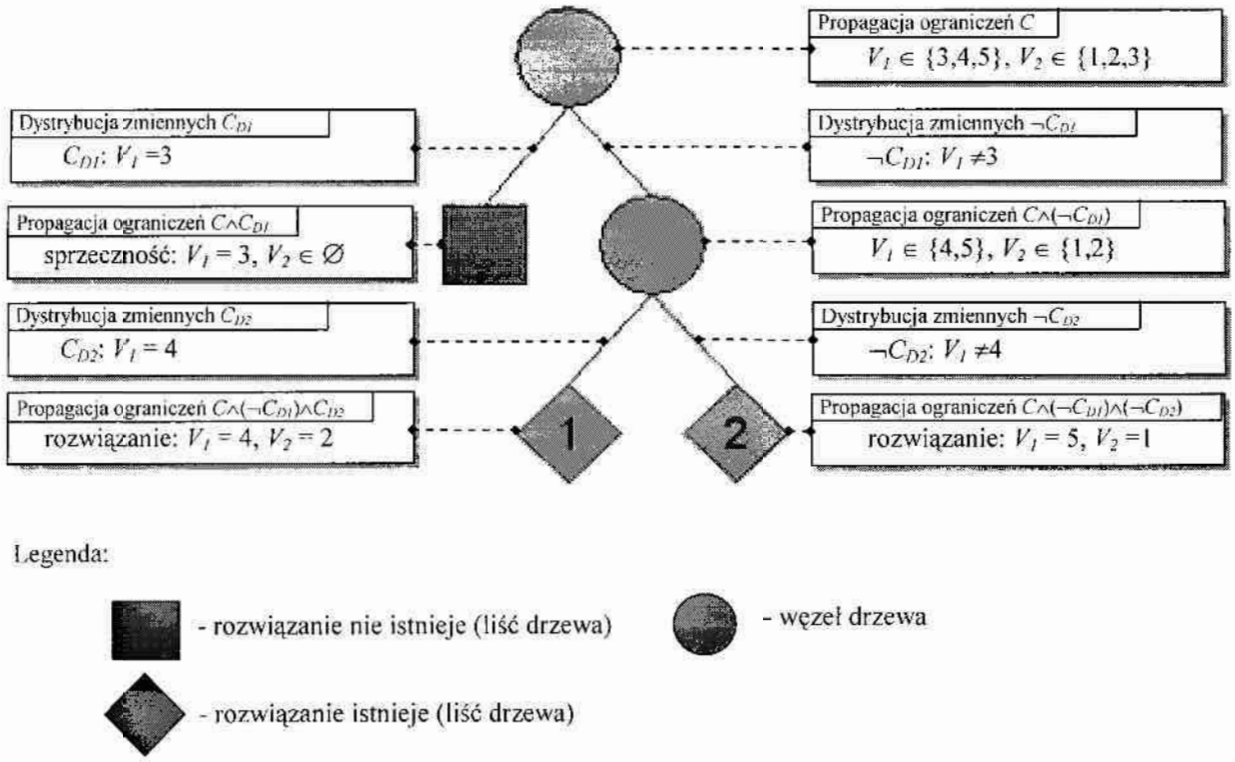
Przedstawiony przykład pokazuje jak efektywny może być proces propagacji ograniczeń (rozmiar przestrzeni został zmniejszony 27 razy) oraz jak ważną rolę odgrywa odpowiedni dobór ograniczeń.

Przykład 3.6. Idea działania dystrybucji zmiennych

Poniższy przykład ilustruje mechanizm dystrybucji zmiennych. Dany jest $PSO = ((V, D), C)$, gdzie zbiór zmiennych decyzyjnych ma postać: $V = \{V_1, V_2\}$, oraz zbiór dziedzin określa dopuszczalne wartości zmiennych V_1, V_2: $D = \{D_1, D_2\}$, $D_1 = \{1,...,5\}$, $D_2 = \{1,...,6\}$ Relacje opisujące związek między zmiennymi określone są zbiorem: $C = \{C_1, C_2\}$, C_1: $V_1 > V_2 + 1$, C_2: $V_1 + V_2 = 6$.

Poszukiwanie rozwiązania spełniającego zadany zbiór ograniczeń sprowadza się do przeszukiwania tak zwanego **drzewa potencjalnych rozwiązań**, które stanowi graficzną reprezentację procesów propagacji ograniczeń i dystrybucji zmiennych. Drzewo potencjalnych rozwiązań odpowiadające rozważanemu przykładowi zostało zilustrowane na rysunku 3.9.

W przedstawionym drzewie wierzchołki reprezentują proces propagacji ograniczeń, gałęzie proces dystrybucji zmiennych, natomiast kształt liści drzewa określa istnienie bądź nie, rozwiązania problemu. W pierwszym kroku w wyniku propagacji ograniczeń dziedziny zmiennych zostały ograniczone do postaci: $V_1 \in \{3,4,5\}$, $V_2 \in \{1,2,3\}$. Ze względu na to, że otrzymane dziedziny są wieloelementowe w dalszej kolejności konieczna była realizacja procesu dystrybucji.



Rys. 3.9. Drzewo potencjalnych rozwiązań ilustrujące proces propagacji ograniczeń i dystrybucji zmiennych

Problem został podzielony na dwa podproblemy. W pierwszym problemie zbiór ograniczeń został uzupełniony o ograniczenie C_{D1}: $V_1 = 3$. W wyniku propagacji ograniczeń tego problemu otrzymano pustą dziedzinę zmiennej V_2 – rozwiązanie nie istnieje. W drugim problemie zbiór ograniczeń został uzupełniony o ograniczenie $\neg C_{D1}$: $V_1 \neq 3$. W wyniku propagacji ograniczeń tego problemu nastąpiło dalsze zawężenie zmiennych, co w kolejnych etapach propagacji i dystrybucji doprowadziło do otrzymania dwóch rozwiązań dopuszczalnych: $V_1 = 4, V_2 = 2$ i $V_1 = 5, V_2 = 1$.

Uzyskane rozwiązania otrzymano po 5 krokach obliczeniowych. Przy czym przez **krok obliczeniowy** rozumiany jest wierzchołek drzewa, odpowiadający jednokrotnej realizacji etapu propagacji i etapu dystrybucji. Innymi słowy jeden krok obliczeniowy oznacza proces ukonkretnienia pojedynczej zmiennej decyzyjnej.

Łatwo zauważyć, że niepusta przestrzeń potencjalnych rozwiązań, uzyskana po procesie propagacji, nie zawsze gwarantuje istnienie rozwiązania (przypadek otrzymania sprzeczności w drzewie z rysunku 3.9).

■

3.2.2 Zalety i ograniczenia zastosowań technik programowania z ograniczeniami

Przy rozwiązywaniu praktycznych problemów istotne jest to by ogólną wiedzę na temat danego problemu umieć wyrazić w postaci zbioru ograniczeń C. Skuteczność metod CP zależy od sposobu sformułowania zbioru ograniczeń. Im zbiór ten jest większy, tym

skuteczność procedur propagacji jest większa. Specyfikacja problemu w postaci ograniczeń charakteryzuje się dużą elastycznością. Jakiekolwiek modyfikacje problemu prowadzą jedynie do konieczności zmiany odpowiedniego ograniczenia i/lub liczby zmiennych oraz zmiany dziedzin zmiennych. Dlatego też techniki *CP* są bardzo chętnie stosowane przy budowie systemów wspomagania decyzji.

Od projektanta systemu wymaga się bardzo dobrej znajomości, tak i istoty problemu, jak i znajomości narzędzi implementacyjnych, które w znacznym stopniu ograniczają możliwości specyfikacji problemu. Zadanym ograniczeniom stawiany jest warunek by w pełni odwzorowywały naturę problemu. Poniżej przedstawiono przykład specyfikacji *PSO* dla problemu składowania z przykładu 3.4.

Przykład 3.7. Problem składowania – zastosowanie programowania z ograniczeniami

Przykład stanowi kontynuację przykładu 3.4. Dla problemu przedstawionego w przykładzie 3.4. należy odpowiedzieć na pytanie: *Czy zadana partia towarów może zostać przewieziona w jednym transporcie? Jeżeli tak, to w jaki sposób ułożyć kontenery w samochodzie?*

W przykładzie 3.4 zaprezentowany został schemat faktów opisujący ogólne zasady charakteryzujące problem. Schemat faktów wykorzystany został do specyfikacji zbioru ograniczeń *C*. Przyjęto założenie, że wszystkie fakty wchodzące w skład schematu F_P są prawdziwe, stąd też zbiór ograniczeń *C* jest postaci :

$$C = \{ Q_{PA1}(x, y, rx, ry, Rx, Ry,L) = \bar{1}, Q_{PA2}(x, y, rx, ry, Rx, Ry,L) = \bar{1}\}.$$

gdzie: $Q_{PA1}(x, y, rx, ry, Rx, Ry,L) = \bar{1}$ – ograniczenie oznacza, że wszystkie fakty zbioru $F_{PA1}(x, y, rx, ry, Rx, Ry,L)$ są zdaniami prawdziwymi: $Q_{A1,1}(x, y, rx, ry, Rx, Ry,L) = 1$, $Q_{A1,2}(x, y, rx, ry, Rx, Ry,L) = 1, \ldots, Q_{A1,at}(x, y, rx, ry, Rx, Ry, L) = 1, \ldots, Q_{A1,at-L}(x, y, rx, ry, Rx, Ry, L) = 1$. Analogicznie $Q_{PA2}(x, y, rx, ry, Rx, Ry,L) = \bar{1}$.

Przyjęty zbiór ograniczeń ma zatem postać (analogicznie do postaci faktów sparametryzowanych):

$C_P = C_{A1} \cup C_{A2}$

Ad. 1) Ograniczenia odpowiadające zasadzie 1:

$C_{A1} = \{C_{1,1}, C_{1,2}, \ldots, C_{1,at}, \ldots, C_{1,at-L}\}$, gdzie: $at = \binom{L}{2}$, L = 5 – liczba kontenerów

$C_{1,r}$: $(x_i \geq x_j + rx'_j) \vee (x_j \geq x_i + rx'_i) \vee (y_i \geq y_j + ry'_j) \vee (y_j \geq y_i + ry'_i) = 1$,

$C_{1,l}$: $[(o_k = 0) \Rightarrow (rx'_k = rx_k) \wedge (rx'_k = rx_k)] \vee [(o_k = 1) \Rightarrow (rx'_k = ry_k) \wedge (ry'_k = rx_k)] = 1$,

gdzie: $i = 1,2, \ldots, L\text{-}1; j = i+1, \ldots, L$; $k = 1,2, \ldots, L$,

o_k – parametr orientacji *k*-tego kontenera,

$$o_k = \begin{cases} 0 \text{ - orientacja wzdłuż krawędzi pionowej} \\ 1 \text{ - orientacja wzdłuż krawędzi poziomej} \end{cases},$$

rx'_k, ry'_k – wielkości pomocnicze.

Ad. 2) Ograniczenia odpowiadające zasadzie 2:

$C_{A2} = \{C_{2,1}, C_{2,2}, \ldots, C_{2,L}\}$,

$C_{2,r}$: $(x_i + rx'_i \leq Rx) \wedge (y_i + ry'_i \leq Ry) = 1$,

gdzie: $i = 1,2, \ldots, L$,

rx'_i, ry'_i – wielkości pomocnicze.

W oparciu o wyznaczony zbiór ograniczeń C sformułowana została postać problemu spełnienia ograniczeń:

$PSO = ((V, D), C)$,

gdzie: $V = V_x \cup V_y \cup V_o$,

$V_x = \{x_1, x_2, \ldots, x_L\}$, $V_y = \{y_1, y_2, \ldots, y_L\}$, $V_o = \{o_1, o_2, \ldots, o_L\}$ – zmienne decyzyjne reprezentujące współrzędne położenia kontenerów oraz ich orientację,

$D = D_x \cup D_y \cup D_o$,

$D_x = \{D_{x,1}, D_{x,2}, \ldots, D_{x,L}\}$, $D_y = \{D_{y,1}, D_{y,2}, \ldots, D_{y,L}\}$, $D_o = \{D_{o,1}, D_{o,2}, \ldots, D_{o,L}\}$ – dziedziny zmiennych decyzyjnych, $D_{x,i} = \{0, 1, \ldots, Rx-1\}$, $D_{y,i} = \{0, 1, \ldots, Ry-1\}$, $D_{o,i} = \{0,1\}$,

$C = C_{A1} \cup C_{A2}$ – przyjęty zbiór ograniczeń na zbiorze zmiennych V_x, V_y, V_o.

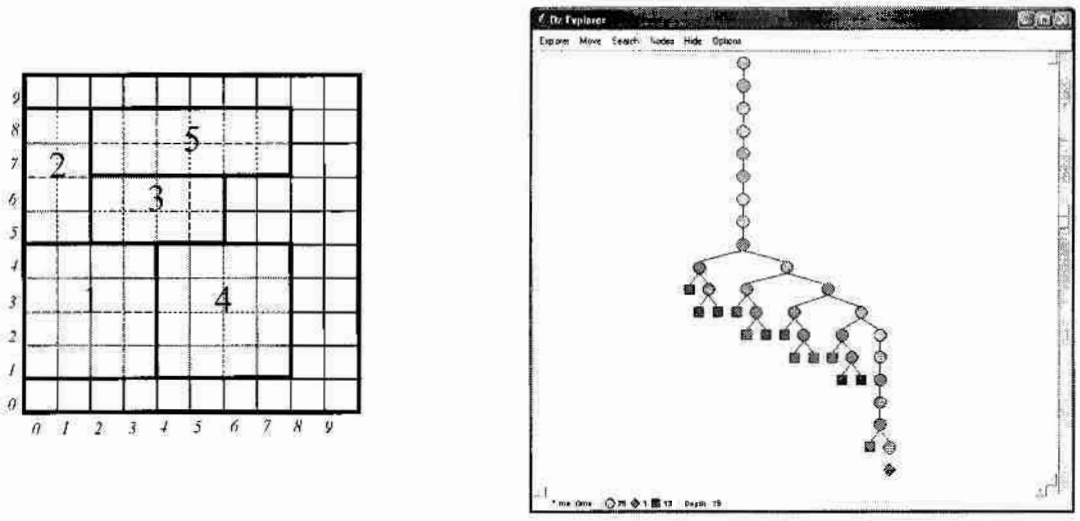
W wyniku rozwiązania powyższego problemu PSO wyznaczane są wartości zmiennych V_x, V_y, V_o, dla których spełnione są przyjęte ogólne zasady rozmieszczania kontenerów. Uzyskane rozwiązanie stanowi odpowiedź na pytanie postawione na wstępie przykładu.

Przykładowe rozwiązanie dopuszczalne (wyznaczone przy użyciu środowiska programowania z ograniczeniami **Oz Mozart**) zostało przedstawione na rysunku 3.10.

Wyznaczone wartości zmiennych decyzyjnych rozwiązania z rysunku 3.10 mają postać:

$V_x = (0, 0, 2, 4, 2)$, $V_y = (1, 5, 5, 1, 7)$, $V_o = (0, 0, 0, 0, 0)$.

Odpowiedź na zadane pytanie jest więc pozytywna, a otrzymane wartości zmiennych określają w jaki sposób ułożyć kontenery w samochodzie Do jego wyznaczenia konieczna była realizacji 27 kroków obliczeniowych.



Rys. 3.10. Rozmieszczenie kontenerów, drzewo potencjalnych rozwiązań uzyskane w środowisku Oz Mozart

■

Przedstawiony problem ilustruje podstawową zaletę technik programowania z ograniczeniami – deklaratywność. Rozwiązanie problemu magazynowania sprowadziło się do specyfikacji odpowiedniego zbioru ograniczeń C i rozwiązania go poprzez zastosowanie rutynowych mechanizmów (propagacji i dystrybucji). Wyznaczony zbiór ograniczeń stanowi reprezentację schematu faktów F_P w terminologii *PSO*.

Analogicznie do schematu faktów, otrzymany zbiór C_p jest sparametryzowanym zbiorem ograniczeń: $C_P = \{C_1(pr), C_2(pr), \ldots, C_r(pr)\}$, gdzie $C_i(pr)$ oznacza i-te ograniczenie, którego postać jest zależna od zbioru parametrów $pr = \{pr_1, pr_2, \ldots, pr_l\}$. W ogólności C_P jest definiowany jako zbiór ograniczeń w postaci sparametryzowanych faktów, którym przypisuje się wartości logiczne 1:

$$C_P = \{ Q_P(u,w,y,pr) = \bar{1}\}, \tag{3.10}$$

gdzie: $Q_P(u,w,y,pr) = \bar{1}$ – ograniczenie oznacza, że wszystkie fakty zbioru $F_P(u,w,y,pr)$ są zdaniami prawdziwymi (wartość logiczna faktów równa jest 1): $Q_{P1}(u, w, y, pr) = 1$, $Q_{P2}(u, w, y, pr) = 1, \ldots, Q_{PK(pr)}(u, w, y, pr) = 1$.

Schematy ograniczeń C_P mogą również być używane do budowy różnych odmian problemów *PSO*. W takich przypadkach schematy ograniczeń C_P stanowią opis cech i własności charakterystycznych dla opisywanych odmian *PSO* co pozwala na sprowadzenie tych problemów do klasycznej postaci *PSO*. Przykładem takich problemów mogą być, tzw. dynamiczne problemy spełniania ograniczeń [53], jak też i inne odmiany rozmytych problemów spełniania ograniczeń [36], [49], [66].

W dalszych rozważaniach przedstawiono sposób w jaki, za pomocą wykorzystania odpowiedniego schematu ograniczeń, można rozwiązywać problemy o rozmytych zmiennych decyzyjnych.

W odróżnieniu od klasycznych propozycji formułowania *PSO* [29], [53], [81] proponowane sformułowanie **Rozmytego Problemu Spełniania Ograniczeń** (***RPSO***) obejmuje.

$$RPSO = ((V_r, D_r), C), \tag{3.11}$$

gdzie: $V_r = \{V_{r,1}, V_{r,2}, \ldots, V_{r,n}\}$, zbiór zmiennych decyzyjnych rozmytych,

$V_{r,i}$ – zmienna rozmyta zdefiniowana w postaci zbioru rozmytego [65]:

$$V_{r,i} = \{(\mu_i(v), v)\}, \quad \forall\, v \in D_{r,i},$$

$\mu_i(v)$ – funkcja przynależności przyporządkowuje każdemu elementowi v pewną wartość ze zbioru [0,1], $\mu_i(v)$: $D_{r,i} \rightarrow [0,1], \forall v \in D_{r,i}$,

$D_r = \{D_{ri} \mid i = 1..n\}$ – rodzina dziedzin zmiennych decyzyjnych, D_{ri} dziedzina i-tej zmiennej decyzyjnej będąca podzbiorem liczb rzeczywistych: $D_{ri} \subseteq R$,

$C = \{C_i \mid i = 1..Lc\}$ – skończony zbiór ograniczeń będący predykatem opisanym na zbiorze zmiennych rozmytych.

Rozwiązaniem tak zdefiniowanego rozmytego problemu spełniania ograniczeń (*RPSO*), jest taka postać rozmytych zmiennych V_r dla których wszystkie ograniczenia ze zbioru C spełnione są w zadanym stopniu pewności.

Opisanie zmiennych V_r poprzez funkcje $\mu(v)$ pozwala na przypisywanie im wartości niepewnych w postaci: „około 4", „wydaje mi się, że około 2", „między 3 a 4", itp. Zbiór ograniczeń jest zbiorem relacji opisujących związki między poszczególnymi zmiennymi rozmytymi, którym dodatkowo przypisany jest określony stopień pewności. Poniżej przedstawiono przykładowe ograniczenia dotyczące zmiennych rozmytych A, B i E

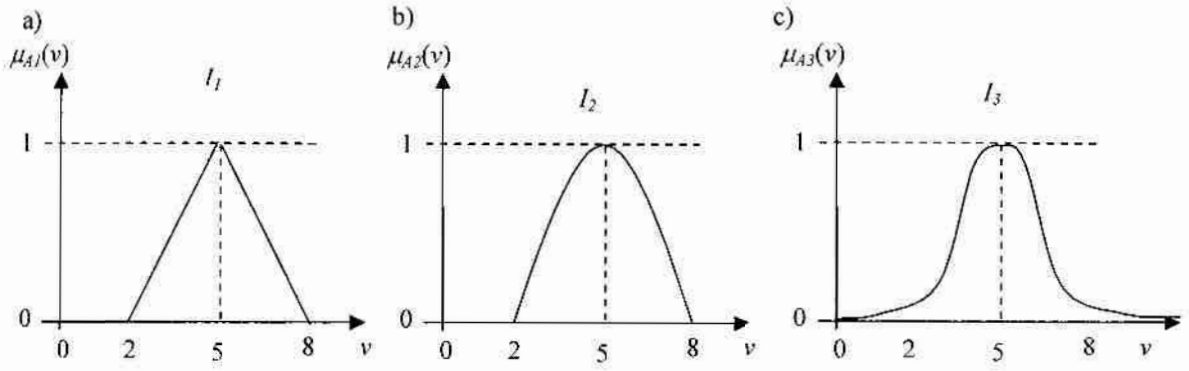
C_1: $(A+B = E) = 1$,
C_2: (E = „około 5") = 1,
C_3: (B = „około 2") = 1.

Wszystkim podanym ograniczeniom nadano stopień pewności 1, jednak w wielu praktycznych przypadkach stopień pewności może być z przedziału [0,1]. Poszukiwanie rozmytej wartości zmiennej A, dla której ograniczenia C_1, C_2, C_3 są spełnione daje wynik w postaci A = „około 3". Ze względu na to, że wyrażenie „około" nie jest jednoznaczne, to tego typu wynik może być interpretowany na wiele sposobów. Liczba interpretacji jest zależna od liczby postaci funkcji przynależności $\mu_A(v)$ (w ogólności jest ona nieskończona). Liczba „około 3" może być wyrażona w postaci funkcji trójkątnej, gaussowskiej, parabolicznej, itp. Zatem poszukiwanie rozwiązania problemu rozmytego *RPSO* może odbywać się tylko w kontekście pewnej przyjętej interpretacji I lub zbioru interpretacji $\{I_1, I_2, \ldots, I_{qn}\}$.

Interpretacja $I_i = \{\mu_{i,1}(v), \mu_{i,2}(v), \ldots, \mu_{i,n}(v)\}$ jest zbiorem funkcji przynależności, gdzie $\mu_{i,j}(v)$ jest funkcją przynależności j-tej zmiennej decyzyjnej w i-tej interpretacji. Określona interpretacja jest zwykle zadawana arbitralnie.

Przykład 3.8. Ilustracja różnych wariantów interpretacji *I*

Przykład ma na celu ilustrację różnych interpretacji I dla określonej zmiennej rozmytej. Dana jest zmienna A o charakterze rozmytym, dane jest ograniczenie określające wartość zmiennej: A jest około 5. Na rysunku 3.11 przedstawione zostały możliwe interpretacje zmiennej A.



Rys. 3.11. Przykładowe interpretacje zmiennej A określające postać funkcji przynależności $\mu_A(v)$, a) funkcja trójkątna, b) paraboliczna, c) gaussowska

Na rysunku 3.11 zostały przedstawione przykładowe funkcje przynależności zmiennej A. Zbiór interpretacji jest postaci: $I = \{I_1, I_2, I_3\}$, gdzie $I_1 = \{\mu_{A1}(v)\}$, $I_2 = \{\mu_{A2}(v)\}$, $I_3 = \{\mu_{A3}(v)\}$. Przedstawione funkcje przynależności odpowiadają kolejno funkcji: ***trójkątnej, parabolicznej*** i ***gaussowskiej***. W zależności od potrzeby zmienna A może być wyrażona w postaci określonej interpretacji.

■

Ze względu na brak metod i narzędzi, które pozwoliłyby na bezpośrednią implementację i rozwiązywanie tego typu problemów spełniania ograniczeń zaproponowano podejście polegające na transformacji *RPSO* do klasycznej postaci *PSO*.

W szczególności, funkcja przynależności $\mu_{i,j}(v)$ należąca do interpretacji I_i jest opisywana przez zbiór parametrów $\{a_{f,1}, a_{f,2}, \ldots, a_{f,rf}\}_{\mu i,j}$, np. funkcja liniowa $\mu_{i,j}(v) = a_{f,1}v + a_{f,2}$ opisywana jest przez parametry $a_{f,1}$, $a_{f,2}$. Parametry $\{a_{f,1}, a_{f,2}, \ldots, a_{f,rf}\}_{\mu i,j}$ określają własności zadanych funkcji przynależności.

Rozwiązaniem rozmytego problemu spełniania ograniczeń są takie postacie zbiorów V_{rj}, które spełniają w zadanym stopniu ograniczenia ze zbioru C. Poszukiwanie zbiorów V_{rj} sprowadza się zatem do wyznaczania postaci funkcji przynależności $\mu_{i,j}(v)$ w kontekście arbitralnie zadanej interpretacji I_i. Jeżeli funkcja $\mu_{i,j}(v)$ może zostać opisana zbiorem parametrów $\{a_{f,1}, a_{f,2}, \ldots, a_{f,rf}\}_{\mu i,j}$, to wyznaczenie postaci zbioru V_{rj} polega na wyznaczeniu wartości parametrów funkcji $\mu_{i,j}(v)$ reprezentującej ten zbiór. Zakładając, że parametry $\{a_{f,1}, a_{f,2}, \ldots, a_{f,rf}\}_{\mu i,j}$ przyjmują tylko wartości dyskretne rozwiązanie rozmytego problemu spełniania ograniczeń sprowadza się do rozwiązania odpowiedniego problemu *PSO* będącego „ostrą" reprezentacją rozważanego problemu.

PSO stanowi opis problemu w kontekście parametrów $\{a_{f,1}, a_{f,2}, \ldots, a_{f,rf}\}_{\mu i,j}$ i zadanej interpretacji. Ze względu na to, że jednemu *RPSO*, w zależności od przyjętej interpretacji I_i, może odpowiadać wiele postaci *PSO*, zatem problemy te oznaczane będą indeksem interpretacji, której odpowiadają: PSO_{Ii}.

Przykład 3.9. Transformacja RPSO do PSO

Przykład ma na celu ilustrację transformacji *RPSO* do klasycznej postaci problemu spełniania ograniczeń *PSO*.

Dany jest rozmyty problem spełniania ograniczeń postaci:

$$RPSO = ((V_r, D_r), C_r),$$

gdzie: $V_r = \{A, B, E\}$ – rozmyte zmienne decyzyjne opisane odpowiednio funkcjami przynależności $\mu_A(v)$, $\mu_B(v)$, $\mu_E(v)$,

$D_r = \{D_{r,A}, D_{r,B}, D_{r,E}\}$ – rzeczywiste dziedziny zmiennych decyzyjnych,

$C = \{C_1, C_2, C_3\}$ – zbiór ograniczeń postaci:

C_1: $(A + B = E) = 1$,

C_2: $(E =$ „około 5”$) = 1$,

C_3: $(B =$ „około 2”$) = 1$,

Przyjęta interpretacja I_1, która zakłada, że zmienne A, B, E, są opisane w postaci funkcji trójkątnej (rysunek 3.12):

$$I_1 = \{\mu_{1,A}(v), \mu_{1,B}(v), \mu_{1,E}(v)\},$$

gdzie:

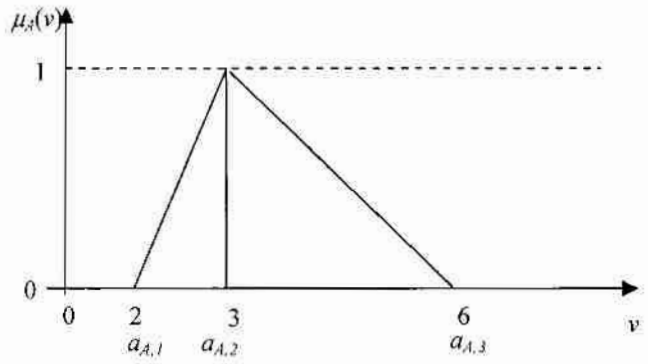
$$\mu_{1,A}(v) = \begin{cases} 0 & \text{dla } v \notin [a_{fA,1}, a_{fA,3}] \\ \dfrac{v}{a_{fA,2} - a_{fA,1}} - \dfrac{a_{fA,1}}{a_{fA,2} - a_{fA,1}} & \text{dla } v \in [a_{fA,1}, a_{fA,2}], \\ \dfrac{-v}{a_{fA,3} - a_{fA,2}} + \dfrac{a_{fA,3}}{a_{fA,3} - a_{fA,2}} & \text{dla } v \in [a_{fA,2}, a_{fA,3}] \end{cases}$$

$\mu_{1,B}(v)$, $\mu_{1,E}(v)$ – definiowane są analogicznie do $\mu_{1,A}(v)$.

W przyjętej interpretacji zdefiniowane funkcje przynależności są opisane parametrami:

$$\mu_{1,A}(v): \{a_{fA,1}, a_{fA,2}, a_{fA,3}\}_{\mu 1,A};\ \mu_{1,B}(v): \{a_{fB,1}, a_{fB,2}, a_{B,3}\}_{\mu 1,B};\ \mu_{1,E}(v): \{a_{fE,1}, a_{fE,2}, a_{fE,3}\}_{\mu 1,E}.$$

Przedstawione parametry opisują trójkątne funkcje przynależności (zgodnie z notacją rysunku 3.12).



Rys. 3.12. Przykładowa trójkątna funkcja przynależności $\mu_A(v)$ opisana parametrami $\{a_{fA,1}, a_{fA,2}, a_{fA,3}\}_{\mu 1,A}$

Problem *RPSO* w kontekście interpretacji I_1, reprezentowany jest przez problem PSO_I definiowany następująco:

$$PSO_{II} = ((a^{II}, D^{II}), C^{II}),$$

gdzie: $a^{II} = a_{fA} \cup a_{fB} \cup a_{fE}$,

$a_{fA} = \{a_{fA,1}, a_{fA,2}, a_{fA,3}\}$, $a_{fB} = \{a_{fB,1}, a_{fB,2}, a_{fB,3}\}$, $a_{fE} = \{a_{fE,1}, a_{fE,2}, a_{fE,3}\}$,

$D^{II} = D_A \cup D_B \cup D_E$, $D_A = \{D_{A,1}, D_{A,2}, D_{A,3}\}$, $D_B = \{D_{B,1}, D_{B,2}, D_{B,3}\}$, $D_E = \{D_{E,1}, D_{E,2}, D_{E,3}\}$.

W przedstawionym problemie zmienne decyzyjne a^{II} są parametrami funkcji przynależności. Dziedziny zmiennych $D_{A,i}$, $D_{B,i}$, $D_{E,i}$ stanowią dyskretne podzbiory dziedzin $D_{r,A}$, $D_{r,B}$, $D_{r,E}$. Ograniczenia $C^{II} = \{C_1^{II}, C_2^{II}, C_3^{II}\}$ stanowią reprezentację ograniczeń $C = \{C_1, C_2, C_3\}$ w kontekście działań na trójkątnych zbiorach rozmytych (przyjętej interpretacji I_1) [65]. Opisują relacje zachodzące między parametrami trójkątnych funkcji przynależności przy wykonywaniu działań arytmetycznych takich jak dodawanie, przypisywanie wartości, itp. Zgodnie z [65] przyjmują one postać:

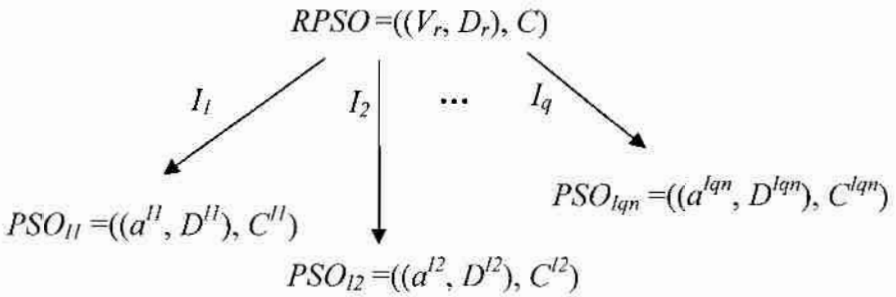
$$C_1^{II} : (a_{fA,1} + a_{fB,1} = a_{fC,1}) \wedge (a_{fA,2} + a_{fB,2} = a_{fC,2}) \wedge (a_{fA,3} + a_{fB,3} = a_{fC,3}) = 1,$$
$$C_2^{II} : (a_{fE,1} < a_{fE,2}) \wedge (a_{fE,2} = 5) \wedge (a_{fE,2} < a_{fE,3}) = 1,$$
$$C_3^{II} : (a_{fB,1} < a_{fB,2}) \wedge (a_{fB,2} = 2) \wedge (a_{fB,2} < a_{fB,3}) = 1.$$

Rozwiązanie problemu PSO_{II} wiąże się z wyznaczeniem takich wartości parametrów a_A, a_B, a_E, które spełniają ograniczenia C^{II}. Znając wartości parametrów a_{fA}, a_{fB}, a_{fE}, możliwe jest zatem, w oparciu o funkcje $\mu_{I,A}(v)$, $\mu_{I,B}(v)$, $\mu_{I,E}(v)$ określenie postaci zbiorów A, B, E, dla których spełnione są ograniczenia C.

■

Przedstawioną na powyższym przykładzie koncepcję transformacji rozmytego problemu spełniania ograniczeń *RPSO* do postaci problemu spełniania ograniczeń PSO_{II}, zilustrowano na rysunku 3.13.

Należy podkreślić, że transformacja problemu *RPSO* do postaci PSO_{Ii} odbywa się przy założeniu, że dla funkcji przynależności wynikających z przyjętej interpretacji istnieją dyskretne parametry a^{Ii} opisujące te funkcje.



Legenda:
RPSO – rozmyty problem *PSO*,
I_i – *i*-ta interpretacja,
PSO_{Ii} – *PSO* odpowiadający interpretacji *Ii*.

Rys. 3.13. Transformacja problemu $RPSO_V$ do problemów *PSO* w kontekście założonego zbioru interpretacji *I*

Przykład 3.10. Problem składowania – podejście rozmyte

Przykład ma na celu ilustrację rozwiązania problemu składowania z przykładu 3.4. przy założeniu, że jest to problem o charakterze rozmytym.

Dany jest problem składowania opisany w przykładzie 3.4. Należy odpowiedzicć na pytanie *Czy zadana partia towarów może zostać przewieziona w jednym transporcie? Jeżeli tak, to w jaki sposób ułożyć kontenery w samochodzie?*

Poszukiwanie odpowiedzi na powyższe pytanie sprowadza się do wyznaczenia współrzędnych położenia kontenerów x_i, y_i, jednak w tym przypadku poszukiwane współrzędne traktowane są jako zmienne rozmyte definiowane w postaci:

$$X_{r,i} = \{(\mu_{r,xi}(v), v)\},\ v \in D_{r,xi},$$
$$Y_{r,i} = \{(\mu_{r,yi}(v), v)\},\ v \in D_{r,yi}.$$

Zazwyczaj w potocznym rozumowaniu położenie kontenerów określamy w sposób nieprecyzyjny np. „umieść paczkę około 0,5 metra od ściany"

Ponadto, w odróżnieniu od przykładu 3.4 przyjęto stałą orientację kontenerów zgodną z rysunkiem 3.4. Pozostałe parametry są takie same jak w przykładzie 3.4.

W rozwiązywanym przypadku ogólne zasady układania kontenerów należy wyrazić w terminologii rozmytego problemu spełniania ograniczeń. Rozważany problem jest postaci:

$$RPSO = ((V_r, D_r), C),$$

gdzie: $V_r = V_{rx} \cup V_{ry}$ – zbiór rozmytych zmiennych decyzyjnych, $V_{rx} = \{X_{r,1}, X_{r,2}, \ldots, X_{r,L}\}$, $V_{ry} = \{Y_{r,1}, Y_{r,2}, \ldots, Y_{r,L}\}$,

$D_r = D_{rx} \cup D_{ry}$ – zbiór dziedzin zmiennych decyzyjnych rozmytych, $D_{rx} = \{D_{X,r1}, D_{X,r2}, \ldots, D_{X,rL}\}$, $D_{X,ri} = [0, Rx]$; $D_{r,y} = \{D_{Y,r1}, D_{Y,r2}, \ldots, D_{Y,rL}\}$, $D_{Y,ri} = [0, Ry]$,

$C = C_{A1} \cup C_{A2}$ – zbiór ograniczeń na zbiorze zmiennych decyzyjnych rozmytych V_{rx}, V_{ry}.

Przyjęty zbiór ograniczeń ma postać:

$C = C_{A1} \cup C_{A2}$,

Ad. 1) Ograniczenia odpowiadające zasadzie 1:

$C_{A1} = \{C_{1,1}, C_{1,2}, \ldots, C_{1,at}\}$, gdzie: $at = \binom{L}{2}$,

$C_{1,r}: (X_{r,i} \geq X_{r,j} + rx_j) \vee (X_{r,j} \geq X_{r,i} + rx_i) \vee (Y_{r,i} \geq Y_{r,j} + ry_j) \vee (Y_{r,j} \geq Y_{r,i} + ry_i) = 1$,

gdzie: $i = 1,2, \ldots, L\text{-}1; j = i+1, \ldots, L$,

Ad. 2) Ograniczenia odpowiadające zasadzie 2:

$C_{A2} = \{C_{2,1}, C_{2,2}, \ldots, C_{2,L}\}$,

$C_{2,r}: (X_{r,i} + rx_i \leq Rx) \wedge (Y_{r,i} + ry_i \leq Ry) = 1$,

gdzie: $i = 1,2, \ldots, L$.

Odpowiedź na wcześniej zadane pytanie sprowadza się do rozwiązania *RPSO*, czyli wyznaczenia takiej postaci rozmytych zmiennych V_{rx}, V_{ry}, dla których ograniczenia C nie są sprzeczne. W tym celu, *RPSO* transformowane jest do postaci PSO_{I1}. Przyjęta interpretacja I_1, zakłada, że zmienne $X_{r,i}$, $Y_{r,i}$, są opisywane za pomocą trójkątnych funkcji przynależności (analogicznie jak w przykładzie 3.9, rysunek 3.12). Funkcje przynależności $\mu_{r,xi}(v)$, $\mu_{r,yi}(v)$, są zatem reprezentowane przez trójki: $\{a_{fX,i1}, a_{fX,i2}, a_{X,i3}\}_{\mu r,xi}$, $\{a_{fY,i1}, a_{fY,i2}, a_{fY,i3}\}_{\mu r,yi}$.

Transformacja *RPSO* do odpowiadającego mu PSO_{I1} determinuje konieczność wyrażenia zbioru ograniczeń C, (określonego dla rozmytych zmiennych decyzyjnych), w postaci zbioru ograniczeń C^{I1} (określonego dla dyskretnych parametrów funkcji przynależności). Dla wyrażeń logicznych wykorzystywanych w ograniczeniach zbioru C przyjęto następującą interpretację.

Stopień spełniania $W_r \in [0,1]$ wyrażenia logicznego postaci: $A \vee B$, gdzie A, B, to funkcje zdaniowe opisane na zmiennych rozmytych, jest wyznaczany z zależności [25]:

$$W_r = W_A + W_B - W_B \cdot W_A,$$

gdzie: W_A, W_B – stopnie spełnienia relacji A, B, $W_A, W_B \in [0,1]$.

Analogicznie stopień spełniania $W_r \in [0,1]$ wyrażenia logicznego postaci: $A \wedge B$, gdzie A, B to relacje opisane na zmiennych rozmytych, jest wyznaczany z zależności [25], [65]:

$$W_r = W_B \cdot W_A,$$

gdzie: W_A, W_B – stopnie spełnienia relacji A, B, $W_A, W_B \in [0,1]$.

Stopień spełnienia relacji A postaci: $(V_{r,1} \leq V_{r,2})$, gdzie: $V_{r,1}$, $V_{r,2}$ to trójkątne zmienne rozmyte opisane parametrami $\{a_{fV,11}, a_{fV,12}, a_{fV,13}\}_{\mu V1}$, $\{a_{fV,21}, a_{fV,22}, a_{fV,23}\}_{\mu V2}$ jest wyznaczany z zależności:

$$W_A = \begin{cases} 0 & \text{gdy}: a_{fV,12} > a_{fV,22} \text{ i } a_{fV,11} > a_{fV,21} \\ \dfrac{a_{fV,23} - a_{fV,11}}{a_{fV,23} - a_{fV,22} + a_{fV,12} - a_{fV,11}} & \text{gdy}: a_{fV,12} > a_{fV,22} \text{ i } a_{fV,11} < a_{fV,21} \\ 1 & \text{gdy}: a_{fV,12} < a_{fV,22} \end{cases}.$$

W oparciu o zdefiniowane zależności sformułowany został zbiór ograniczeń C^{II}. Przyjęte ograniczenia opisują relacje zachodzące między parametrami trójkątnych funkcji przynależności:

$$C^{II} = \{ C^{II}_{1,r1}, C^{II}_{1,r2}, .., C^{II}_{1,at16}, \ldots C^{II}_{2,1,2} \}.$$

Ad.1) Ograniczenia określające stopień spełnienia $Wx_{i,j}$ wyrażenia $(X_{r,i} \geq X_{r,j} + rx_j)$:

$$C^{II}_{1,r1}: \left[(a_{fX,j2} > a_{fX,i2}) \wedge (a_{fX,j1} > a_{fX,i1}) \Rightarrow (Wx_{i,j} = 0) \right] = 1 .$$

$$C^{II}_{1,r2}: \left[(a_{fX,i2} > a_{fX,j2}) \wedge (a_{fX,j1} < a_{fX,i1}) \Rightarrow \left(Wx_{ij} = \frac{a_{fX,j3} - (a_{fX,j1} + rx_j)}{a_{fX,i3} - a_{fX,i2} + a_{fX,j2} - a_{fX,j1}} \right) \right] = 1 .$$

$$C^{II}_{1,r4}: \left[(a_{fX,j2} < a_{fX,i2}) \Rightarrow (Wx_{i,j} = 1) \right] = 1 .$$

Ograniczenia określające stopień spełnienia Wx_{ji} wyrażenia $(X_{rj} \geq X_{ri} + rx_i)$:

$$C^{II}_{1,r5}: \left[(a_{fX,i2} > a_{fX,j2}) \wedge (a_{fX,i1} > a_{fX,j1}) \Rightarrow (Wx_{j,i} = 0) \right] = 1 .$$

$$C^{II}_{1,r6}: \left[(a_{fX,i2} > a_{fX,j2}) \wedge (a_{fX,i1} < a_{fX,j1}) \Rightarrow \left(Wx_{j,i} = \frac{a_{fX,j3} - (a_{fX,i1} + rx_i)}{a_{fX,j3} - a_{fX,j2} + a_{fX,i2} - a_{fX,i1}} \right) \right] = 1 .$$

$$C^{II}_{1,r7}: \left[(a_{fX,i2} < a_{fX,j2}) \Rightarrow (Wx_{j,i} = 1) \right] = 1 .$$

Ograniczenia określające stopień spełnienia Wy_{ji} wyrażenia $(Y_{ri} \geq Y_{rj} + ry_j)$:

$$C^{II}_{1,r8}: \left[(a_{fY,j2} > a_{fY,i2}) \wedge (a_{fY,j1} > a_{fY,i1}) \Rightarrow (Wy_{i,j} = 0) \right] = 1 .$$

$$C^{II}_{1,r9}: \left[(a_{fY,i2} > a_{fY,j2}) \wedge (a_{fY,j1} < a_{fY,i1}) \Rightarrow \left(Wy_{i,j} = \frac{a_{fY,j3} - (a_{fY,j1} + ry_j)}{a_{fY,i3} - a_{fY,i2} + a_{fY,j2} - af_{Y,j1}} \right) \right] = 1 .$$

$$C^{II}_{1,r10}: \left[(a_{fY,j2} < a_{fY,i2}) \Rightarrow (Wy_{i,j} = 1) \right] = 1 .$$

Ograniczenia określające stopień spełnienia Wy_{ji} wyrażenia $(Y_{rj} \geq Y_{ri} + ry_i)$:

$$C^{II}_{1,r11}: \left[(a_{fY,i2} > a_{fY,j2}) \wedge (a_{fY,i1} > a_{fY,j1}) \Rightarrow (Wy_{j,i} = 0) \right] = 1 .$$

$$C^{II}_{1,r12}: (a_{fY,j2} > a_{fY,j2}) \wedge (a_{fY,i1} < a_{fY,j1}) \Rightarrow \left(Wy_{j,i} = \frac{a_{fY,j3} - (a_{fY,j1} + ry_i)}{a_{fY,j3} - a_{fY,j2} + a_{fY,i2} - a_{fY,i1}} \right).$$

$$C^{II}_{1,r13}: \left[(a_{fY,j2} < a_{fY,j2}) \Rightarrow (Wy_{j,i} = 1) \right] = 1 .$$

Ograniczenia określające stopień spełnienia Wx wyrażenia $(X_{r,i} \geq X_{r,j} + rx_j) \vee (X_{r,j} \geq X_{r,i} + rx_i)$:

$C^{II}_{1,r14}$: $Wx_{i,j} + Wx_{j,i} - Wx_{i,j} \cdot Wx_{j,i} = Wx$.

Ograniczenia określające stopień spełnienia Wy wyrażenia $(Y_{r,i} \geq Y_{r,j} + ry_j) \vee (Y_{r,j} \geq Y_{r,i} + ry_i)$

$C^{II}_{1,r15}$: $Wy_{i,j} + Wy_{j,i} - Wy_{i,j} \cdot Wy_{j,i} = Wy$.

Ograniczenia reprezentujące ograniczenie $C_{1,r}$:

$C^{II}_{1,r16}$: $Wx + Wy - Wy \cdot Wx = 1$,

gdzie: $i = 1,2, \ldots, L\text{-}1$; $j = i+1, \ldots, L$.

Ad.2) Ograniczenia reprezentujące ograniczenia $C_{2,r}$:

$C^{II}_{2,r1}$: $[(a_{fX,i1} + rx_i \leq Rx) \wedge (a_{fX,i2} + rx_i \leq Rx) \wedge (a_{fX,i3} + rx_i \leq Rx)] = 1$,

$C^{II}_{2,r2}$: $[(a_{fY,i1} + ry_i \leq Ry) \wedge (a_{fY,i2} + ry_i \leq Ry) \wedge (a_{fY,i3} + ry_i \leq Ry)] = 1$.

Przedstawiony zbiór ograniczeń został wykorzystany do sformułowania problemu PSO_{II}, który ma postać:

$$PSO_{II} = ((a^{II}, D^{II}), C^{II}),$$

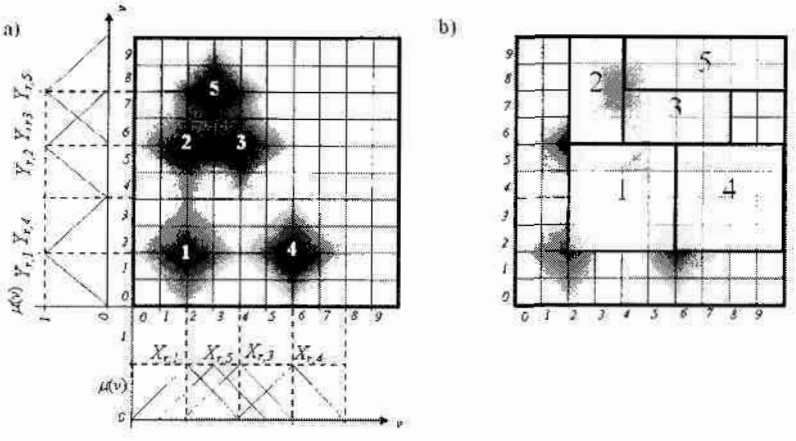
gdzie: $a^{II} = a_{fX} \cup a_{fY}$,

$a_{fX} = \{a_{fX,11}, a_{fX,12}, a_{fX,13}, a_{fX,21}, \ldots, a_{fX,L3}\}$, $a_Y = \{a_{fY,11}, a_{fY,12}, a_{fY,13}, a_{fY,21}, \ldots, a_{fY,L3}\}$ – zbiory zmiennych decyzyjnych,

$D = D_X \cup D_Y$ – zbiór dziedzin zmiennych decyzyjnych, $D_X = \{D_{X,11}, D_{X,12}, D_{X,13}, D_{X,21}, \ldots, D_{X,L3}\}$, $D_{X,ij} = \{0,1, \ldots, Rx\text{-}1\}$, $D_Y = \{D_{Y,11}, D_{Y,12}, D_{Y,13}, D_{Y,21}, \ldots, D_{Y,L3}\}$, $D_{Y,ij} = \{0,1, \ldots, Ry\text{-}1\}$,

C^{II} – zbiór przyjętych ograniczeń.

Rozwiązując problem PSO_I wyznaczane są takie wartości parametrów a_{fX}, a_{fY}, które spełniają ograniczenia C^{II}. Wartości parametrów a_{fX}, a_{fY}, pozwalają określić kształty funkcji przynależności poszukiwanych rozmytych zmiennych decyzyjnych V_{rx}, V_{ry}. Przykładowe rozwiązanie dopuszczalne zostało przedstawione na rysunku 3.14.



Rys. 3.14. Przestrzeń ładunkowa: a) rozmyte obszary rozmieszczenia kontenerów, b) rozmieszczenie kontenerów zgodnie ze wskazanymi obszarami

Obszary reprezentujące zaciemnione pola na powierzchni ładunkowej określają miejsce położenia poszczególnych kontenerów. W przeciwieństwie do przykładu 3.4 wyznaczone współrzędne położenia kontenerów nie są wartościami ostrymi, dlatego też prezentowane są

graficznie jako rozmyte pola, których kształt jest konsekwencją złożenia dwóch trójkątnych funkcji przynależności. Dla przykładu zmiennym rozmytym $X_{r,1}$, $Y_{r,1}$, odpowiada obszar oznaczony na rysunku 3.14 a) numerem 1. Intensywność koloru przedstawionych obszarów oznacza stopień przynależności punktów powierzchni ładunkowej do zmiennych $X_{r,1}$, $Y_{r,1}$ (kolor czarny oznacza stopień przynależności - 1, kolor biały - 0).

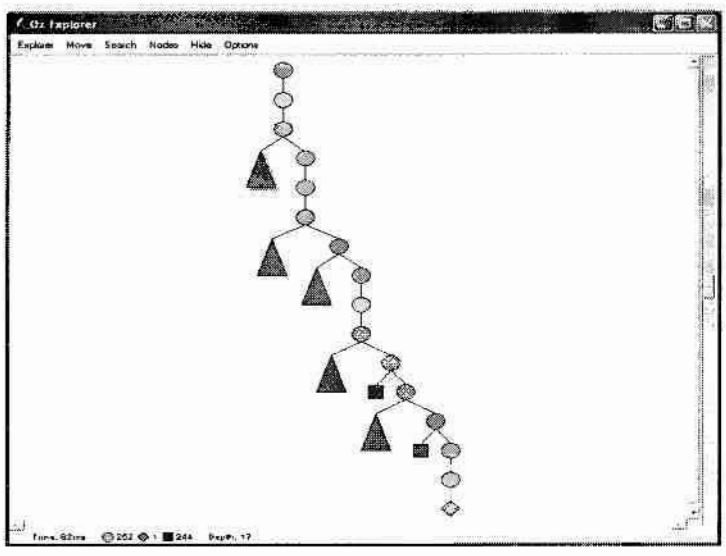
Ułożenie kontenerów zgodnie z rysunkiem 3.14 a) tak, że lewy dolny róg każdego z kontenerów umieszczony jest w centrum odpowiadającym jego obszarowi, gwarantuje (stopień pewności 1), że spełnione będą przyjęte ogólne zasady rozmieszczania (kontenery nie nachodzą na siebie i nie przekraczają granic powierzchni ładunkowej). Ustawienie kontenerów w każdym innym miejscu, np. w obszarze o odcieniu szarości, zmniejsza pewność (stopień pewności z przedziału (0,1)), spełnienia przedstawionych ogólnych zasad.

Rozwiązanie zostało otrzymane w wyniku implementacji problemu PSO_1 w środowisku programowania z ograniczeniami **Oz Mozart**. Drzewo potencjalnych rozwiązań zostało przedstawione na rysunku 3.15. Rozwiązanie w postaci parametrów funkcji przynależności zostało wyznaczone w wyniku realizacji 497 kroków obliczeniowych. Kolejno dla zmiennych rozmytych $X_{r,1}$, $X_{r,2}$, $X_{r,3}$, $X_{r,4}$, $X_{r,5}$, parametry mają postać:

$$\{0,2,4\}_{\mu r,x1};\ \{0,2,4\}_{\mu r,x1};\ \{2,4,6\}_{\mu r,x1};\ \{4,6,8\}_{\mu r,x1};\ \{1,3,5\}_{\mu r,x1}.$$

Kolejno dla zmiennych rozmytych $Y_{r,1}$, $Y_{r,2}$, $Y_{r,3}$, $Y_{r,4}$, $Y_{r,5}$, parametry mają postać:

$$\{0,2,4\}_{\mu r,y1};\ \{4,6,8\}_{\mu r,y1};\ \{4,6,8\}_{\mu r,y1};\ \{0,2,4\}_{\mu r,y1};\ \{6,8,10\}_{\mu r,y1}.$$



Rys. 3.15. Drzewo potencjalnych rozwiązań uzyskane w środowisku Oz Mozart. Rozwiązane dopuszczalne otrzymane po 497 krokach

■

W przykładzie 3.10 przedstawiono sposób budowy ograniczeń dla klasycznej postaci problemu *PSO*, tak by móc rozwiązywać równoważny mu *RPSO*. Zdefiniowany zbiór ograniczeń stanowi więc schemat ograniczeń C_P, który odwzorowuje relacje zachodzące

między zmiennymi rozmytymi w postaci relacji dyskretnych parametrów funkcji trójkątnych. Podobnie jak w przykładzie 3.7. zbiór ograniczeń jest sparametryzowany, a więc można go swobodnie stosować w innych problemach magazynowania.

Budowa **schematów ograniczeń** pozwala uniknąć konieczności formułowania nowych ograniczeń za każdym razem gdy należy rozwiązać problemy należące do tej samej klasy. Zatem raz zbudowany schemat ograniczeń może być wykorzystywany wielokrotnie w różnych problemach należących do tej samej klasy.

Do głównych zalet technik programowania z ograniczeniami zalicza się możliwość deklaratywnej specyfikacji problemów, (możliwe jest budowanie opisów w postaci schematów ograniczeń) oraz wykorzystania jednolitego mechanizmu rozwiązywania opartego na procedurach propagacji i dystrybucji zmiennych.

Wadą tych technik jest ograniczenie się tylko do przestrzeni zmiennych dyskretnych, przez co w wielu przypadkach (np. problemy rozmyte) należy wykorzystywać (jeśli to możliwe) do rozwiązywania tego typu problemów, schematy ograniczeń (umożliwiających transformację problemu do klasycznej postaci *PSO*).

3.3. Implementacja metody logiczno–algebraicznej w technikach programowania z ograniczeniami

Reprezentację wiedzy $KB = <U, W, Y; Re>$ można przedstawić w postaci problemu spełniania ograniczeń *PSO* [16] (rysunek 3.16). W przypadku problemu *PSO* odwzorowującego reprezentację wiedzy *KB*, rolę ograniczeń *C* spełniają fakty wchodzące w skład zbioru $F(u,w,y)$ (charakteryzujących postać relacji *Re*), przy czym przyjmowane jest, że wszystkie fakty są zdaniami prawdziwymi. Rolę zmiennych *V* spełniają zmienne *u*, *y*, *w*. Dziedziny zmiennych są określone w postaci zbiorów D_u, D_w, D_y. Problem *PSO* jest definiowany następująco:

$$PSO = ((V, D), \{Q(u,w,y) = \bar{1}\}), \tag{3.12}$$

gdzie: $V = Vu \cup Vw \cup Vy$,

$Vu = \{u_1, u_2, \ldots, u_{ku}\}$, $Vw = \{w_1, w_2, \ldots, w_{kw}\}$, $Vy = \{y_1, y_2, \ldots, y_{ky}\}$ – zbiory zmiennych wejściowych, pomocniczych i wyjściowych, reprezentacji wiedzy *KB*.

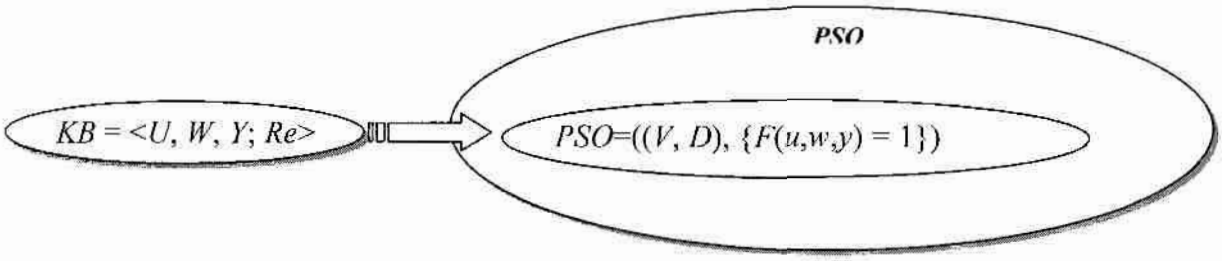
$D = D_u \cup D_w \cup D_y$,

$D_u = \{D_{u,1}, D_{u,2}, \ldots, D_{u,ku}\}$, $D_{u,i} = U$, $D_w = \{D_{w,1}, D_{w,2}, \ldots, D_{w,kw}\}$, $D_{w,i} = W$, $D_y = \{D_{y,1}, D_{y,2}, \ldots, D_{y,ky}\}$, $D_{y,i} = Y$ – zbiory wartości dyskretnych zmiennych decyzyjnych,

$Q(u,w,y) = \bar{1}$ – oznacza ograniczenie, że wszystkim faktom zbioru $F(u,w,y)$ przypisywana jest wartość logiczna 1: $Q_1(u,w,y) = 1$, $Q_2(u,w,y) = 1$, ... , $Q_K(u,w,y) = 1$.

Rozwiązaniem tak rozumianego problemu *PSO* jest zbiór wartości zmiennych *u*, *y*, *w*, dla których prawdziwe są wszystkie ograniczenia przedstawiane w postaci zdań logicznych $F(u,w,y)$. Przedstawione przykłady 3.4 i 3.10 ilustrują wykorzystanie takiego przekształcenia do poszukiwania odpowiedzi na pytanie dotyczące rozmieszczenia kontenerów

w ograniczonej przestrzeni ładunkowej. Należy zauważyć, że w obu przykładach końcowe postacie zbiorów ograniczeń (np. po przekształceniu rozmytego problemu *RPSO* do postaci PSO_{II}) mają postać zdań logicznych. Sformułowane problemy *PSO*, stanowią więc reprezentację wiedzy (*KB*) zadaną w postaci ograniczeń.



Rys. 3.16. Reprezentacja wiedzy *KB* jako problem *PSO*

W rozważanych przykładach nie było konieczne przedstawianie problemu najpierw jako reprezentacji wiedzy (*KB*), następnie wyrażania go w terminach problemu spełniania ograniczeń. Specyfikację problemu można rozpocząć od razu od formułowania postaci ograniczeń. Ze względu na „prostotę" problemów etap formułowania ogólnej wiedzy w postaci faktów (zasad układania kontenerów) mógł zostać pominięty. Jako odpowiedź na zadane pytanie należało rozwiązać odpowiedni problem *PSO*. W wielu problemach jednak sama specyfikacja w postaci ograniczeń nie jest wystarczająca.

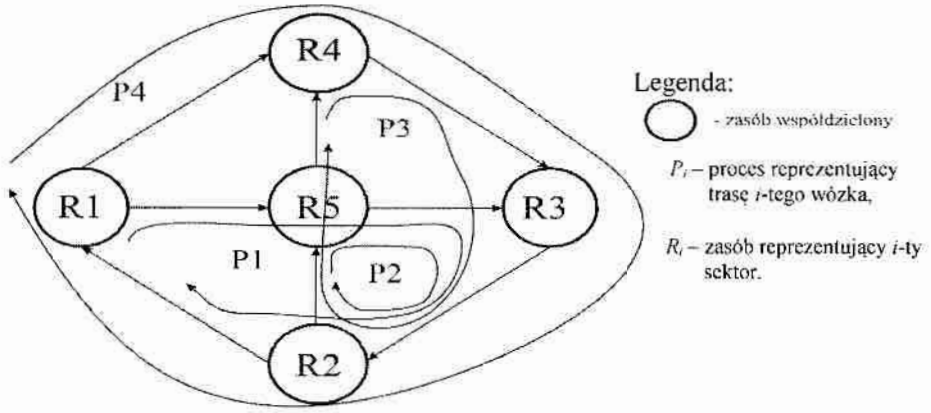
W większości przypadków, aby móc wykorzystać postać problemów *PSO* i technik programowania z ograniczeniami *CP* w systemach wspomagania decyzji należy posiadać wiedzę na temat warunków wystarczających, spełnienie których gwarantuje, że w przestrzeni potencjalnych rozwiązań istnieją rozwiązania dopuszczalne. Warto zauważyć, że istnieją problemy, dla których nie jest znana a priori postać zbioru ograniczeń, pozwalająca na rozwiązywanie problemu bez znajomości odpowiednich warunków wystarczających. Do takich problemów należą problemy harmonogramowania pracy wózków samojezdnych w systemach transportowych.

Przykład 3.11. Problem harmonogramowania w systemach transportowych

Dany jest system transportowy zawierający poruszające się wózki i zasoby. System jest charakteryzowany przez graf (rysunek 3.17) reprezentujący strukturę połączeń między zasobami i marszruty realizacji operacji na określonych zasobach przez poszczególne wózki [50].

Procesy P_1, P_2, P_3, P_4, reprezentują ruch poszczególnych wózków, wózki poruszają się pomiędzy zasobami R_1, R_2, R_3, R_4, R_5, wzdłuż określonych tras. Wózki realizują swoje operacje cyklicznie. Sekwencja $P_i = (R_o, R_p, \ldots, R_r)$ określa kolejność zasobów, przez które przejeżdża wózek. Każdy zasób R_i opisany jest wektorem dostępu procesów do zasobu $\sigma_i = (P_j, P_l, \ldots, P_r)$, którego elementy określają kolejność obsługi procesów przez *i*-ty zasób np. $\sigma_1 = (P_1, P_4)$ oznacza, że na R_1 jako pierwszy obsługiwany jest proces P_1 a następnie P_4, $\sigma_5 = (P_2, P_1, P_3)$ oznacza, że jako pierwszy na R_5 obsługiwany jest proces P_1 a następnie P_2

i P_3. Wózki poruszają się zgodnie z marszrutami, wózek może rozpocząć ruch od dowolnego zasobu marszruty. Przemieszczanie się między zasobami odbywa się w czasie $T = 0$, czas obsługi wózka na określonych zasobach opisany jest wektorem T_i. Wózki obsługiwane są na określonych zasobach zgodnie z kolejnością obsługi wózków σ_i.



Rys. 3.17. Graf dla systemu transportowego wózków samojezdnych

Dla tak sformułowanych założeń stawiane jest zwykle pytanie: *Czy istnieje harmonogram pracy wózków, gwarantujący realizację wszystkich procesów tak by w systemie nie wystąpiła blokada i/lub kolizja wózków?*

Poszukiwane są zatem rozwiązania, które nie prowadzą do blokady częściowej bądź całkowitej systemu. Przez pojęcie blokady (w kolejnym rozdziale pojęcie to będzie uszczegółowione) rozumie się stan systemu, w którym niemożliwa jest dalsza praca co najmniej jednego wózka samojezdnego.

Dla przedstawionego problemu znane są specyfikacje *PSO* [29], [81], [79], zakładające znajomość wszystkich warunków, determinujących wartości parametrów systemu, które gwarantują brak występowania blokady. Warunki tego typu stanowią właściwości, których spełnienie gwarantuje, że w rozważanym systemie transportowym istnieje odpowiedź na postawione powyżej pytanie. Posiadanie wiedzy o tych właściwościach jest niezbędne do rozwiązania problemu. Poszukiwanie warunków odbywa się zgodnie z procedurą przedstawioną w rozdziale 2 (rozwiązanie problemu decyzyjnego). Dlatego też konieczne wydaje się najpierw reprezentowanie wiedzy o problemie w postaci *KB*, a następnie wyrażenie reprezentacji wiedzy w postaci odpowiedniego problemu *PSO*.

■

Możliwość wykorzystania metody logiczno-algebraicznej do specyfikacji wiedzy *KB* w postaci problemów spełniania ograniczeń (*PSO*), umożliwia wykorzystanie technik programowania z ograniczeniami. Okazuje się, że również problem wyznaczania warunków wystarczających sprowadza się do rozwiązania odpowiedniego problemu decyzyjnego. Właściwość reprezentacji *KB* w postaci *PSO* może być wykorzystana zatem do rozwiązywania problem decyzyjnego. W rozważanym przypadku wyznaczenie zbiorów S_{ul}

i S_{u2} niezbędnych do określenia postaci faktu $Fu(u)$ (rozumianego jako warunek gwarantujący spełnienie faktu $Fy(y)$) sprowadza się do rozwiązania następujących problemów *PSO*:

$$\text{Dla } S_{u1}: \quad PSO_{Su1} = ((V, D), \{Q(u,w,y) = \bar{1}, Qy(y) = \bar{1}\}), \tag{3.13}$$

$$\text{Dla } S_{u2}: \quad PSO_{Su2} = ((V, D), \{Q(u,w,y) = \bar{1}, Qy(y) = \bar{0}\}), \tag{3.14}$$

gdzie: $V = Vu \cup Vw \cup Vy$, $Vu = \{u_1, u_2, \ldots, u_{ku}\}$, $Vw = \{w_1, w_2, \ldots, w_{kw}\}$, $Vy = \{y_1, y_2, \ldots, y_{ky}\}$ – zbiory zmiennych wejściowych, pomocniczych i wyjściowych, reprezentacji wiedzy *KB*

$D = D_u \cup D_w \cup D_y$,

$D_u = \{D_{u,1}, D_{u,2}, \ldots, D_{u,ku}\}$, $D_{u,i} = U$, $D_w = \{D_{w,1}, D_{w,2}, \ldots, D_{w,kw}\}$, $D_{w,i} = W$, $D_y = \{D_{y,1}, D_{y,2}, \ldots, D_{y,ky}\}$, $D_{y,i} = Y$ – zbiory wartości dyskretnych zmiennych decyzyjnych,

$Q(u,w,y) = \bar{1}$ – ograniczenie oznacza, że wszystkie fakty zbioru $F(u,w,y)$ są zdaniami prawdziwymi: $Q_1(u,w,y) = 1$, $Q_2(u,w,y) = 1$, ..., $Q_K(u,w,y) = 1$,

$Qy(y) = \bar{1}$ – ograniczenie oznacza, że wszystkie fakty zbioru $F_y(y)$ są zdaniami prawdziwymi: $Qy_1(y) = 1$, $Qy_2(y) = 1$, ..., $Qy_{Rf}(y) = 1$,

$Qy(y) = \bar{0}$ – ograniczenie oznacza, że co najmniej jeden fakt zbioru $Fy(y)$ jest zdaniami fałszywym.

W zdefiniowanych problemach PSO_{Su1}, PSO_{Su2}, fakty $F(u,w,y)$ i $Fy(y)$ opisują relacje miedzy poszczególnymi zmiennymi decyzyjnymi. Zgodnie z założeniem, że wiedza wchodząca w skład reprezentacji *KB* jest prawdziwa, faktom (określającym postać relacji *Re*) $F(u,w,y)$ przypisuje się wartości 1. Problemy PSO_{Su1}, PSO_{Su2} stanowią reprezentację zdefiniowanych układów równań (3.5), (3.6). Wyrażenie układów równań w postaci problemów spełniania ograniczeń pozwala na wykorzystanie technik programowania z ograniczeniami do rozwiązania tych układów. Podejście to stanowi alternatywę do metod opartych na analizie tablicy prawdy i metod dekompozycji.

Okazuje się jednak, że klasyczne metody poszukiwania rozwiązań (oparte na dystrybucji wszystkich zmiennych i propagacji ograniczeń) bywają często, w kontekście tych problemów, niewystarczające. Dlatego też powstaje konieczność poszukiwania strategii efektywnego przeszukiwania drzewa potencjalnych rozwiązań.

3.4. Podsumowanie

System interakcyjnego wspomagania decyzji powinien posiadać wiedzę w postaci warunków wystarczających, która gwarantować będzie, że na zadane pytanie użytkownika istnieje odpowiedź i poszukiwanie tej odpowiedzi ma sens. Poszukiwanie warunków wystarczających odpowiada niejako poszukiwaniu właściwości wejściowych bazy wiedzy, których spełnienie gwarantuje spełnienie zadanych przez użytkownika właściwości wyjściowych (relacji dostarczonych do systemu).

Zaproponowane podejście, oparte na metodzie logiczno-algebraicznej, umożliwia formalizację posiadanej wiedzy i dostarcza mechanizmy wnioskowania pozwalające na

wyznaczanie tego typu właściwości. W przypadku problemów należących do tej samej klasy istnieje możliwość ich opisu w postaci sparametryzowanych zdań zwanych schematami faktów. W oparciu o jeden schemat faktów możliwe jest automatyczne generowanie różnych wariantów bazy wiedzy. Cechą charakterystyczną metody logiczno-algebraicznej jest możliwość wyrażenia jej problemów (problem decyzyjny, problem analizy) w terminologii technik programowania z ograniczeniami.

Techniki programowania z ograniczeniami umożliwiają rozwiązywanie problemów *PSO*, które są powszechnie wykorzystywane do rozwiązywania praktycznych problemów kombinatorycznych. Przy specyfikacji problemów w postaci *PSO* istotna jest postać zbioru ograniczeń. W oparciu o schematy ograniczeń możliwe jest rozwiązywanie problemów pierwotnie nie należących do klasy problemów *PSO*, takich jak rozmyte problemy spełniania ograniczeń itp. W wielu problemach schemat ograniczeń jest konsekwencją transformacji faktów opisujących problem do postaci ograniczeń. Wymagane jest by specyfikacja problemu w postaci odpowiedniego schematu ograniczeń dawała gwarancję istnienia rozwiązania. Możliwość generowania takiej specyfikacji jest uwarunkowana posiadaniem wiedzy w postaci warunków wystarczających. Zgodnie z metodą logiczno-algebriczną poszukiwanie takiej wiedzy odbywa się poprzez rozwiązanie odpowiedniego problemu decyzyjnego.

4. Weryfikacja bazy wiedzy

Podstawowym wymogiem pracy systemu wspomagania decyzji w trybie interakcyjnym jest konieczność posiadania warunków gwarantujących istnienie odpowiedzi na zadany zbiór pytań rutynowych. Przez takie warunki rozumiane są właściwości obiektu (relacje), które gwarantują, że w obiekcie spełnione będą właściwości (relacje) zadane przez decydenta. Pozyskiwanie tego typu warunków odbywa się w procesie weryfikacji bazy wiedzy. Weryfikacja bazy wiedzy polega na zbadaniu jej spójności. Zgodnie z metodą logiczno-algebraiczną sprawdzenie spójności bazy wiedzy polega na stwierdzeniu czy pomiędzy zadanymi właściwości wejściowymi $Fu(u)$ i wyjściowymi $Fy(y)$ istnieje związek logiczny. Inaczej mówiąc, czy w kontekście zadanych faktów $F(u,w,y)$ opisujących obiekt spełniona jest implikacja $Fu(u) \Rightarrow Fy(y)$?

Decydent formując pytanie określa zwykle swoje oczekiwania poprzez sformułowanie postaci właściwości wyjściowej $Fy(y)$. Aby baza wiedzy była spójna pod kątem zadanego pytania, dla właściwości $Fy(y)$ musi istnieć niepusty zbiór właściwości wejściowych $Fu(u)$. Oznacza to, że badanie spójność bazy wiedzy polega na wyznaczeniu właściwości $Fu(u)$. Poszukiwanie właściwości wejściowej, dla znanej właściwości wyjściowej i faktów $F(u,w,y)$ opisujących obiekt, sprowadza się do rozwiązania omówionego w poprzednim rozdziale problemu decyzyjnego. W efekcie otrzymywane są fakty $Fu(u)$, które traktowane są jako warunki gwarantujące istnienie odpowiedzi na zadane pytanie. Podsumowując, weryfikacja bazy wiedzy sprowadza się do rozwiązania odpowiednio sformułowanego problemu decyzyjnego.

Pojęcie spójności jest różnie opisywane w literaturze [1], [42], [96], w ogólnym przypadku można powiedzieć, że dany zbiór zdań logicznych np. $F(u,w,y)$, tworzących pewną bazę wiedzy, jest spójny jeżeli między jego elementami istnieje związek umożliwiający wyznaczenie, przy użyciu odpowiedniego algorytmu, elementów tego zbioru. W rozważanym przypadku można uznać, że zbiór $F(u,w,y) \cup Fu(u)$, będący częścią reprezentacji wiedzy KB, jest spójny z pewnym zbiorem faktów $Fy(y)$ jeśli ze zbioru $F(u,w,y) \cup Fu(u)$ można wykazać (na drodze wnioskowania) prawdziwość (bądź nie) $Fy(y)$. Przedstawione ujęcie jest zgodne z opisanym powyżej podejściem, w którym przez spójność rozumie się związek o charakterze implikacyjnym ($Fu(u) \Rightarrow Fy(y)$) między właściwościami wejściowymi $Fu(u)$ i wyjściowymi $Fy(y)$. Ściślej spójność jest scharakteryzowana w postaci poniższej definicji.

Definicja 4.1.

Niesprzeczny zbiór faktów $F(u,w,y) \cup Fu(u)$ jest spójny z dodatkowym zbiorem $Fy(y)$ jeżeli z faktów $F(u,w,y) \cup Fu(u)$ można wyznaczyć na drodze wnioskowania tylko $Fy(y)$ albo $\neg Fy(y)$.

Weryfikacja spójności bazy wiedzy może polegać na sprawdzeniu czy zadany zbiór $F(u,w,y) \cup Fu(u)$ jest spójny z zadanym $Fy(y)$ lub na poszukiwaniu takiej postaci własności $Fu(u)$ która zagwarantuje spójność $F(u,w,y) \cup Fu(u)$ z zadanym $Fy(y)$. Drugi przypadek jest

szczególnie atrakcyjny w kontekście systemów wspomagania decyzji. Własność $Fu(u)$ w tym kontekście rozumiana jest jako zbiór warunków wystarczających spójności bazy wiedzy. Wykorzystanie problemu decyzyjnego do weryfikacji spójności jest również zgodne z przyjętą definicją. Wynikiem problemu decyzyjnego jest taka postać $Fu(u)$, która gwarantuje spełnienie $Fy(y)$ (dla spełnionego $Fu(u)$ niedopuszczalny jest $\neg Fy(y)$).

Proces weryfikacji bazy wiedzy odbywa się w kontekście przyjętej reprezentacji wiedzy. Z kolei formułowanie postaci wiedzy poprzez określenie postaci schematów faktów, realizowane jest na etapie projektowania systemów wspomagania decyzji. Istotne jest zatem aby sformułowana postać faktów odpowiadała w pełni relacjom opisującym funkcjonowanie modelowanego obiektu, a w szczególności umożliwiała wyznaczenie warunków wystarczających. Innymi słowy system powinien dysponować taką reprezentacją wiedzy *KB*, która pozwala na prowadzenie procesu wnioskowania. Spełnienie tego wymogu jest zadaniem trudnym i ściśle zależnym od natury obiektu.

W rozdziale tym proces tworzenia bazy wiedzy oraz jej weryfikacji zilustrowany został na przykładzie systemu transportowego (dopuszczającego zachodzenie blokad). Dla przyjętego sposobu reprezentacji wiedzy podane zostały sposoby budowy efektywnych czasowo strategii wyznaczania warunków wystarczających.

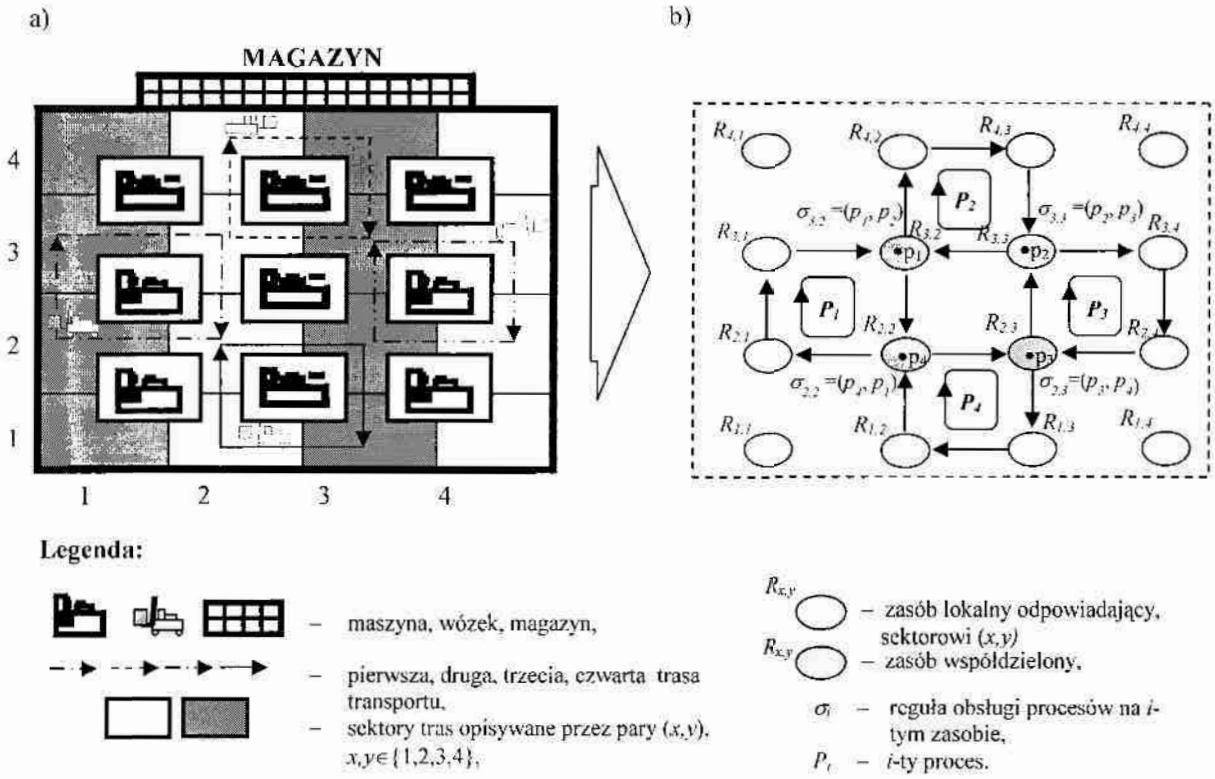
4.1. Badanie spójności bazy wiedzy

Rozważana klasa systemów transportowych może być modelowana w kategoriach **Systemów Współbieżnych Procesów Cyklicznych** (SWPC) [50], [51], [19] [20] , co ilustruje rysunek 4.1.

W systemie klasy SWPC wyróżnia się:

- $R = \{R_1, R_2, \ldots, R_{ks}\}$ – zbiór k - zasobów, odpowiadających kolejnym sektorom, po których poruszają się wózki,
- $P = \{P_1, P_2, \ldots, P_q\}$ – zbiór q - procesów, gdzie każdy proces reprezentuje marszrutę określonego wózka poruszającego się w systemie,
- $P_i = (R_o, R_p, \ldots, R_r)$ – sekwencja, której elementy (zasoby) określają marszrutę i-tego procesu, realizującego kolejne operacje, proces może rozpoczynać się od dowolnej operacji,
- $S_0 = (R^{s0}{}_1, R^{s0}{}_2, \ldots, R^{s0}{}_q)$ – stan początkowy, sekwencja, której elementy określają zasoby, od których kolejne procesy $P_1, \ldots, P_q$, rozpoczynają pracę, $R^{s0}{}_1 \in P_1$, $R^{s0}{}_2 \in P_2, \ldots, R^{s0}{}_q \in P_q$,
- $\Theta = (\sigma_i, \sigma_j, \ldots, \sigma_l)$ – reguły priorytetowania, sekwencja, której elementy σ_i określają reguły obsługi ruchu na zasobach współdzielonych R_i. Przez zasób współdzielony rozumie się zasób występujący co najmniej w dwóch marszrutach P_j i P_j, zasób lokalny jest zasobem, który występuje dokładnie w jednej marszrucie P_k. $\sigma_i = (P_j, P_l, \ldots, P_r)$ – i-ta reguła obsługi ruchu, sekwencja określająca kolejność realizacji procesów na i-tym zasobie współdzielonym,

- $po = (p_{1,1}, p_{1,2}, \ldots, p_{1,m1}, p_{2,1}, p_{2,2}, \ldots, p_{2,m2}, \ldots, p_{q,1}, p_{q,2}, \ldots, p_{q,mq})$ – sekwencja operacji, której elementy reprezentują operacje realizowane w systemie, gdzie p_{ij} oznacza j-tą operację elementarną i-tego procesu,
- $x = (x_1, x_2, \ldots, x_n)$ – harmonogram pracy, jest sekwencją, której elementy x_i określają terminy rozpoczęcia operacji reprezentowanych w sekwencji p przez i-te współrzędne, tzn. x_1 – określa termin rozpoczęcia operacji $p_{1,1}$, x_2 – określa termin operacji $p_{1,2}$, x_n – termin operacji $p_{q,mq}$,
- $t = (t_1, t_2, \ldots, t_n)$ – sekwencja czasów, której elementy t_i określają czasy trwania operacji odpowiadające operacjom o terminie rozpoczęcia x_i.

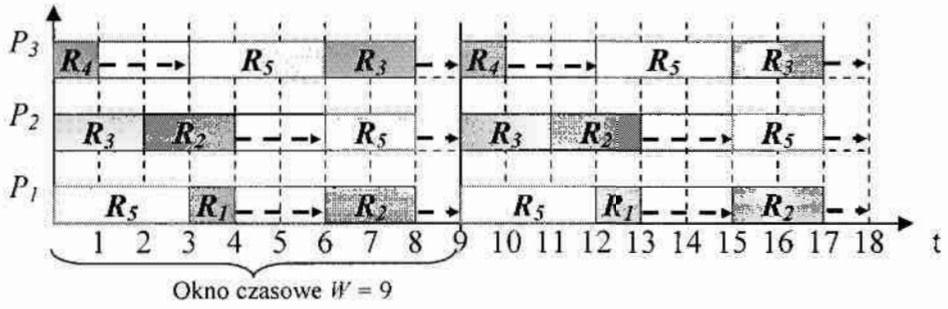


Rys. 4.1. System transportowy ESP a) ilustracja poglądowa , b) model SWPC

Przyjęto, że rozważana klasa systemów spełnia następujące założenia:

1. Realizowane procesy są jednokierunkowe, tzn. realizowane zgodnie z kierunkiem marszruty transportowej P_i.
2. W marszrutach transportowych nie występują nawroty, tzn. w marszrucie procesu P_i określony zasób R_i może wystąpić co najwyżej raz.
3. Operacje realizowane na tym samym zasobie wzajemnie się wykluczają, tzn. w danej chwili na danym zasobie może być wykonywany tylko jeden proces.
4. Procesy są realizowane cyklicznie w zadanych oknach czasowych W. Jeden cykl procesu oznacza pojedyncze wykonanie wszystkich operacji wskazanych przez marszrutę procesu. Na rysunku 4.2 zilustrowano przykład okna czasowego W. W oknie

trwającym 9 jednostek czasu każdy proces: P_1, P_2, P_3, jest realizowany dokładnie jeden raz. Jeśli procesy zakończą pracę przed 9 jednostką czasu to czekają na rozpoczęcie kolejnego okna na zasobie, na którym ukończyły pracę.



Rys. 4.2. Diagram Gantt'a ilustrujący wykorzystanie procesów P.

5. Zasób, który występuje w marszrutach, co najmniej dwóch procesów nazywany jest zasobem współdzielonym. Kolejność cyklicznych realizacji procesów na zasobie współdzielonym jest zadana przez regułę obsługi σ_i. Reguła ta jest definiowana jako ciąg procesów, które kolejno realizowane są na zasobie współdzielonym R_i, np.: $\sigma_i = (P_1, P_2, P_3)$.
6. Proces, który oczekuje na dostęp do zasobu współdzielonego z chwilą jego zwolnienia natychmiast rozpoczyna swoją operację (oczywiście o ile jakiś inny proces nie wyprzedza go w ustalonej uprzednio kolejności). W trakcie oczekiwania na dostęp do zasobu proces przebywa na zasobie, w którym ukończył pracę.

Przedstawione założenia stanowią ogólną wiedzę określającą pewną klasę systemu transportowego. W przypadkach praktycznych spotkać można mniej lub bardziej złożone opisy systemów. Spotykane uszczegółowienia dotyczą, na przykład, możliwości jednoczesnej realizacji wielu procesów na jednym zasobie, możliwości przerywania operacji na poszczególnych zasobach, itp.

Rozważany system transportowy opisywany jest następującą reprezentacją wiedzy:

$$KB = <O, S0, X, PD; Re> , \tag{4.1}$$

gdzie: $O, S0, X, PD$ – zbiory sekwencji Θ, S_0, x, P_D, $\Theta \in O, S_0 \in S0, x \in X, P_D \in PD$,

$\{\Theta, S_0, x, P_D\}$ – zbiór sekwencji opisujących system transportowy; gdzie P_D – sekwencja zmiennych pomocniczych (np. wynikających z ograniczeń implementacyjnych),

$Re = \{\Theta, S_0, x, P_D: Q(\Theta, S_0, x, P_D) = \bar{1}\}$ – relacja określająca wartości elementów sekwencji Θ, S_0, x, P_D,

$Q(\Theta, S_0, x, P_D)$ – ciąg wartości logicznych faktów zbioru $F(\Theta, S_0, x, P_D)$,

$F(\Theta, S_0, x, P_D)$ – zbiór faktów opisujących zasady funkcjonowania systemu, tzn. zasady realizacji operacji poszczególnych procesów, kolejności operacji, ograniczenia zasobowe, zasady określające liczbę obsługiwanych procesów itp.

Przyjęto, że zbiór faktów ma postać:

$$F(\Theta, S_0, x, P_D) = F_a(\Theta, S_0, x, P_D) \cup F_b(\Theta, S_0, x, P_D),$$

gdzie:

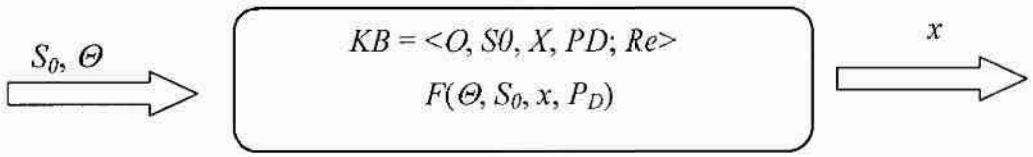
$F_a(\Theta, S_0, x, P_D)$ – zbiór faktów opisujących ogólne zasady panujące w systemie (zgodnie z przyjętymi założeniami rozważanej klasy systemów), tzn. fakty gwarantujące realizację operacji procesu zgodnie z przyjętymi marszrutami, fakty opisujące zachowanie się procesów na zasobach współdzielonych, itp. Fakty F_a charakteryzują zasady wspólne dla wszystkich systemów rozważanej klasy.

$F_b(\Theta, S_0, x, P_D)$ – zbiór faktów dodatkowych opisujący charakterystyczne, indywidualne cechy rozważanego systemu, np. fakty mówiące o tym, że pewien proces musi rozpocząć się w określonym czasie po ukończeniu operacji innego procesu, fakty określające stałą kolejność obsługi procesów na zasobie współdzielonym, itp.

Cechą charakterystyczną przedstawionych założeń jest to, że postać faktów F_a jest wspólna dla każdego systemu transportowego rozważanej klasy, natomiast F_b jest różna i zależna od jego indywidualnych właściwości.

W oparciu o tak zdefiniowaną bazę wiedzy *KB* można prowadzić proces wnioskowania, którego celem jest wyznaczanie warunków gwarantujących istnienie odpowiedzi na zbiór pytań rutynowych. W ogólności, pytania te mogą dotyczyć dowolnej właściwości systemu opisanej sekwencjami Θ, S_0, x, P_D. Konstrukcja rozważanych pytań zwykle umożliwia postrzeganie reprezentacji wiedzy *KB* jako układu wejście/wyjście. Znajomość właściwości opisujących wejście (lub wyjście) układu umożliwia poszukiwanie właściwości wyjściowych (lub wejściowych) układu.

Przyjęto szczególny przypadek gdy wejściem są wartości zmiennych sekwencji S_0, Θ, a wyjściem układu jest wartość zmiennych sekwencji x (rysunek 4.3):



Rys. 4.3. *KB* jako układ wejście/wyjście

Tego typu podział umożliwia, w oparciu o właściwości sekwencji S_0, Θ, wnioskowanie o właściwościach harmonogramu x i odwrotnie, w oparciu o znane właściwości harmonogramu x, możliwe jest wnioskowanie o właściwościach sekwencji S_0, Θ. Właściwości wejściowe i wyjściowe są reprezentowane odpowiednio przez zdania logiczne wejściowe $Fu(S_0, \Theta)$ i zdania logiczne wyjściowe $Fy(x)$.

W takim podejściu, na etapie wnioskowania należy odpowiedzieć na pytanie:

Czy istnieje postać sekwencji S_0 i Θ, które gwarantują, że otrzymany harmonogram x odpowiada bezkolizyjnej i bezblokadowej realizacji procesów?

Poszukiwanie odpowiedzi na powyższe pytanie jest równoznaczne z badaniem spójności bazy wiedzy. Pytanie sprowadza się do wyznaczenia warunków, w postaci zdań

logicznych $Fu(S_0, \Theta)$ opisujących relacje między elementami sekwencji S_0 i Θ, które gwarantują że na wyjściu układu otrzymany zostanie harmonogram bezblokadowy i bezkolizyjny. W przedstawionym ujęciu poszukiwana właściwość $Fu(S_0, \Theta)$ stanowi jednocześnie warunek gwarantujący spójność bazy wiedzy $KB = <O, S0, X, PD; Re>$ z $Fy(x)$.

W kontekście przyjętej postaci układu wejście/wyjście oznacza to wyznaczenie łańcucha wyrażeń logicznym łączący wejście z wyjściem układu, tak że na wyjściu otrzymana (spełniona) zostanie żądana właściwość.

Weryfikacja bazy wiedzy realizowana jest zgodnie z procedurą przedstawioną na rysunku 4.4.

Rys. 4.4. Procedura weryfikacji bazy wiedzy

Procedura obejmuje kolejno etapy:

1. Wyrażenie założeń charakteryzujących klasę rozważanego systemu transportowego, w postaci relacji opisujących związki między zmiennymi modelu SWPC.
2. Wcharakteryzowanie, w kontekście zmiennych modelu SWPC, zadanej przez decydenta, właściwości wyjściowej (w tym przypadku stan blokady i kolizji).
3. Wyrażenie wyznaczonych relacji w postaci schematu faktów.
4. Wozwiązanie problemu decyzyjnego w oparciu o reprezentację wiedzy *KB* (otrzymanej ze schematu faktów) – wyznaczenie warunków wystarczających *Fu*.

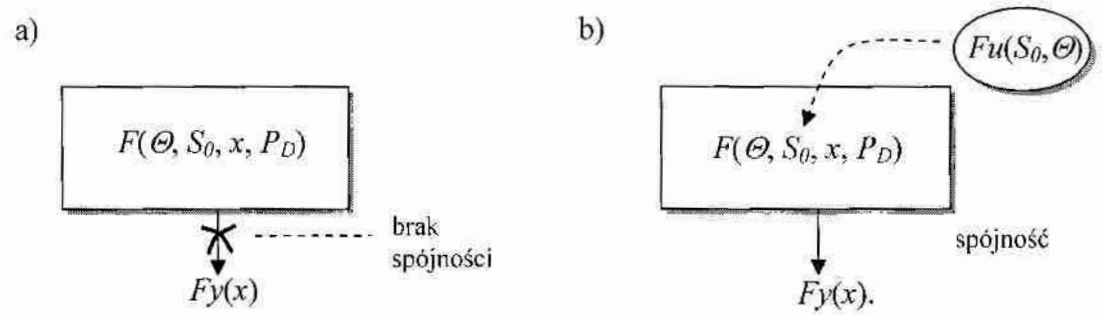
Zgodnie z przedstawioną procedurą weryfikacja bazy wiedzy obejmuje etapy przekształcania jej z postaci opisowej do zbioru faktów i reprezentacji wiedzy oraz etap wyznaczania warunków wystarczających. Wyznaczenie niepustego zbioru warunków wystarczających $Fu(S_0, \Theta)$ oznacza spójność bazy wiedzy pod kątem zadanego pytania. O ile

złożoność obliczeniowa pierwszych trzech etapów, przedstawionej procedury, jest trudna do określenia ze względu na niedeterministyczny charakter tych etapów, o tyle etap ostatni polegający na rozwiązaniu problemu decyzyjnego charakteryzuje się złożonością wykładniczą.

Następne rozdziały opisują kolejno realizację opisanych powyżej zagadnień, na przykładzie obiektów należących do rozważanej klasy systemów transportowych.

4.2. Warunki gwarantujące spójność bazy wiedzy

Na rysunku 4.5a) przedstawiony został przypadek, w którym zbiór $F(\Theta, S_0, x, P_D)$ nie jest spójny z właściwością $Fy(x)$. Jest to typowa sytuacja spotykana w praktyce rzadko się zdarza by zadane przez użytkownika ograniczenia (wyrażone w postaci $Fy(x)$) były spełnione bez względu na możliwe wartości zmiennych Θ, S_0. Może jednak istnieć dodatkowy zbiór $Fu(S_0, \Theta)$, który dodany do zbioru $F(\Theta, S_0, x, P_D)$ gwarantuje, że otrzymany zbiór $F(\Theta, S_0, x, P_D) \cup Fu(S_0, \Theta)$ będzie spójny z $Fy(x)$ (rysunek 4.5b)). Zbiór $Fu(S_0, \Theta)$ w ogólnym przypadku nazywany jest warunkiem gwarantującym spójność bazy wiedzy.



Rys. 4.5. Przykład: a) braku spójności zbioru $F(\Theta, S_0, x, P_D)$ z $Fy(x)$, b) spójności $F(\Theta, S_0, x, P_D) \cup Fu(S_0, \Theta)$ z $Fy(x)$

W przedstawionym rozdziale w kolejnych punktach zgodnie z procedurą przedstawioną na rysunku 4.4 omówione zostało podejście wyznaczania warunków $Fu(S_0, \Theta)$. Warunki te dla rozważanego systemu transportowego są interpretowane jako warunki gwarantujące istnienie odpowiedzi na zadane pytania.

4.2.1. Struktura reprezentacji wiedzy w systemie współbieżnych procesów cyklicznych

W poniższym punkcie przedstawiono sposób formułowania założeń, dotyczących rozważanej klasy systemów transportowych, w postaci ograniczeń określających związki między zmiennymi modelu SWPC.

Ogólne zasady opisujące zachowanie się procesów w systemie SWPC, wyrażane są w postaci zbioru faktów F_a. Fakty F_a reprezentują na poziomie logicznym przyjęte poniżej ograniczenia sposobu realizacji procesów P:

Kolejność realizacji poszczególnych operacji. Kolejność operacji realizowanych w systemie jest określona przez założenia 1 i 2 (realizacja jednokierunkowa i bez nawrotów). W modelu SWPC sekwencje reguł priorytetowania Θ i stanu początkowego procesów S_0, determinują porządek realizacji poszczególnych operacji p, a zatem mają wpływ na terminy x rozpoczęcia realizacji poszczególnych operacji.

Przyjęta postać stanu początkowego S_0 determinuje terminy x rozpoczęcia operacji p w obrębie marszrut realizowanych procesów P. Operacje procesu opisanego marszrutą P_i realizowane są kolejno na zasobach wskazanych przez marszrutę, rozpoczynając od zasobu wskazanego przez i-ty element stanu S_0. Kolejność realizacji operacji jest wyrażana poprzez terminy ich rozpoczęcia x. Związek między stanem początkowym S_0, a terminami rozpoczęcia operacji x przedstawia układ (4.2):

Dla $S_0 = (R^{s0}{}_1, R^{s0}{}_2, \ldots, R^{s0}{}_q)$, $P_i = (R_o, R_p, \ldots, R_r)$, $i = 1,2, \ldots, q$, elementy sekwencji $x^i{}_s = (x_{i,1}, x_{i,2}, \ldots, x_{i,mi})$ spełniają następujący układ nierówności.

$$\begin{cases} x^*_{i,j} = 0 \\ x_{i,j-1} \geq x^*_{i,j} + t_{i,j} \\ x_{i,j-2} \geq x_{i,j+1} + t_{i,j+1} \\ \ldots \\ x_{i,mi} \geq x_{i,mi-1} + t_{i,mi-1} \\ x_{i,1} \geq x_{i,mi} + t_{i,mi} \\ \ldots \\ x_{i,j-1} \geq x_{i,j-2} + t_{i,j-2} \end{cases} \tag{4.2}$$

gdzie: $x^i{}_s$ – sekwencja terminów $x_{i,j}$ rozpoczęcia operacji realizowanych przez i-ty proces na zasobach wskazanych przez marszrutę P_i; $x_{i,j}$ – termin rozpoczęcia operacji $p_{i,j}$; sekwencja $x^i{}_s$ zbudowana jest z elementów sekwencji x,

$m_i = ||P_i||$ – liczba zasobów wchodzących w skład marszruty P_i,

$x^*_{i,j}$ – termin rozpoczęcia operacji i-tego procesu na zasobie $R^{s0}{}_i$ (element sekwencji S_0), $x^*_{i,j}$ jest elementem sekwencji $x^i{}_s$, j – współrzędna terminu rozpoczęcia operacji $x^*_{i,j}$ w sekwencji $x^i{}_s$,

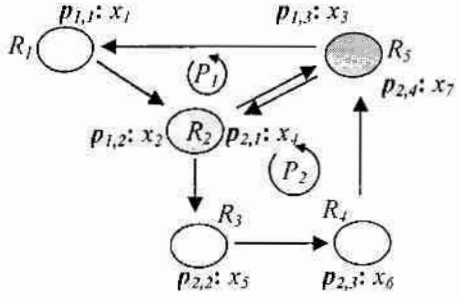
$t_{i,j}$ – czas realizacji operacji $p_{i,j}$.

Intuicja przedstawionego ograniczenia jest następująca: stan początkowy S_0 determinuje zasób $R^{s0}{}_i$, a tym samym operację $x^*_{i,j}$, od której i-ty proces rozpoczyna swoją pracę. Kolejne operacje i-tego procesu powinny rozpoczynać się nie wcześniej niż terminy ukończenia operacji poprzednich. W ogólności, dla każdego procesu formułowany jest oddzielny układ nierówności (4.2). Przedstawione ograniczenie sprowadza się więc do tego by elementy harmonogramu x spełniały q-elementowy zbiór układów nierówności (4.2).

Przykład 4.1. Kolejność obsługi operacji należących do wspólnych marszrut

Celem przykładu jest sformułowanie układu nierówności (4.2) dla dwóch procesów realizowanych w systemie przedstawionym na rysunku 4.6.

Dany jest system transportowy z rysunku 4.6. W systemie realizowane są cyklicznie dwa procesy: $P = \{P_1, P_2\}$, $P_1 = (R_1, R_2, R_5)$, $P_2 = (R_2, R_3, R_4, R_5)$. W jednym cyklu realizowanych jest 7 operacji (każdy proces wykonuje jednokrotnie operacje odpowiadające przyjętej marszrucie): $po = (p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{2,3}, p_{2,4})$. Sekwencji p odpowiada harmonogram pracy $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. Przyjęto następujący stan początkowy: $S_0 = (R_5, R_3)$.



Legenda:

R_i – i-ty zasób lokalny,

R_i – i-ty zasób współdzielony,

$p_{i,j}$ – j-ta operacja elementarna realizowana przez proces P_i,

P_i – i-ty proces,

x_i – termin rozpoczęcia i-tej operacji elementarnej.

Rys. 4.6. Przykład systemu SWPC

Dla każdego procesu układ nierówności (4.2) przyjmuje postać:

Dla P_1:

$$\begin{cases} x_3 = 0 \\ x_1 \geq x_3 + t_3 \\ x_2 \geq x_1 + t_1 \end{cases};$$

Dla P_2:

$$\begin{cases} x_5 = 0 \\ x_6 \geq x_5 + t_5 \\ x_7 \geq x_6 + t_6 \\ x_4 \geq x_7 + t_7 \end{cases}.$$

Dla stanu początkowego S_0 proces P_1 rozpoczyna pracę od zasobu R_5 czyli operacji $p_{1,3}$. Zgodnie z przyjętą zasadą, termin x_3 (odpowiadający operacji $p_{1,3}$) jest równy zero (odpowiada chwil początkowej). Kolejne operacje $p_{1,1}$, $p_{1,2}$, procesu P_1 mogą się rozpocząć nie wcześniej niż terminy ukończenia operacji poprzedzających. Operacja $p_{1,1}$ może rozpocząć się nic wcześniej niż termin ukończenia operacji $p_{1,3}$ stąd: $x_1 \geq x_3 + t_3$. Podobnie operacja $p_{1,2}$, nie może się rozpocząć wcześniej niż termin ukończenia operacji $p_{1,1}$ stąd: $x_2 \geq x_1 + t_1$. Dla procesu P_2 układ nierówności budowany jest analogicznie. Spełnienie przez elementy harmonogramu x sformułowanych układów nierówności gwarantuje, że procesy będą realizowane zgodnie z przyjętą, w marszrutach, kolejnością.

■

Podobnie reguła obsługi ruchu σ_j (wchodząca w skład sekwencji Θ) determinuje kolejność operacji p realizowanych przez procesy P na określonym zasobie współdzielonym R_j. Innymi słowy, reguła σ_j wpływa na wartości elementów x, które odpowiadają operacjom realizowanym w obrębie jednego zasobu współdzielonego. Związek między σ_j, a x przedstawia układ (4.3).

Dla $\Theta = (\sigma_o, \sigma_p, \ldots, \sigma_l)$, $\sigma_j = (P_i, P_l, \ldots, P_r)$, gdzie: $j = 1,2, \ldots, \|\Theta\|$, elementy sekwencji $x^j_\sigma = (x_{1,j}, x_{2,j}, \ldots, x_{ei,j})$ spełniają następujący układ nierówności:

$$\begin{cases} x_{i+1,j} \geq x^{\sigma_j}{}_i + t_{i,j} \\ x_{i+2,j} \geq x_{i+1,j} + t_{i+1,j} \\ \ldots \\ x_{ej,j} \geq x_{ej-1,j} + t_{ej-1,j} \\ x_{1,j} \geq x_{ej,j} + t_{ej,j} \\ \ldots \\ x_{i-1,j} \geq x_{i-2,j} + t_{i-2,j} \end{cases}, \qquad (4.3)$$

gdzie: $e_j = \|\sigma_j\|$ – liczba procesów (realizowanych na zasobie współdzielonym R_j) wchodzących w skład j-tej reguły obsługi ruchu σ_j,

$x^j{}_\sigma$ – sekwencja terminów rozpoczęcia operacji realizowanych przez procesy wchodzące w skład σ_j na zasobie współdzielonym R_j; $x_{i,j}$ – termin rozpoczęcia i-tego procesu (należącego do sekwencji σ_j) na zasobie współdzielonym R_j; sekwencja $x^j{}_\sigma$ zbudowana jest z elementów sekwencji x,

$x^{\sigma_j}{}_i$ – termin rozpoczęcia operacji pierwszego procesu sekwencji σ_j (i-tego w sekwencji $x^j{}_\sigma$) na zasobie R_j, $x^{\sigma_j}{}_i$ jest elementem sekwencji $x^j{}_\sigma$,

$t_{i,j}$ – czas realizacji operacji, reprezentowanej przez termin rozpoczęcia $x_{i,j}$.

Intuicja przedstawionego ograniczenia jest następująca: Rozważane są procesy realizowane na wspólnym zasobie R_j. Reguła σ_j determinuje kolejność realizacji procesów. Operacja pierwszego procesu opisanego terminem $x^{\sigma_j}{}_i$ powinna być realizowana przed operacją drugiego procesu, stąd: $x_{i+1,j} \geq x^{\sigma_j}{}_i + t_{i,j}$. Drugi proces reguły σ_j, opisany terminem $x_{i+1,j}$, powinien być realizowany przed procesem trzecim, stąd: $x_{i+2,j} \geq x_{i+1,j} + t_{i+1,j}$. Analogicznie budowane są ograniczenia dla pozostałych procesów co prowadzi do układu nierówności (4.3).

W ogólności, dla każdego zasobu współdzielonego formułowany jest układ nierówności (4.3). Przedstawione ograniczenie sprowadza się więc do tego by elementy harmonogramu x spełniały q-elementowy zbiór układów (4.3).

Przykład 4.2. Kolejność obsługi operacji na zasobach współdzielonych

Celem przykładu jest sformułowanie układu nierówności (4.3) dla systemu z rysunku 4.6.

Dla rozważanego w poprzednim przykładzie systemu (z rysunku 4.6) przyjęto następujące reguły priorytetowania: $\Theta = (\sigma_2, \sigma_5)$, $\sigma_2 = (P_1, P_2)$, $\sigma_5 = (P_1, P_2)$. Reguły określają kolejność obsługi procesów na zasobach współdzielonych R_2 i R_5. Dla zadanych reguł terminy rozpoczęcia operacji na tych zasobach muszą spełniać następujące nierówności (zgodnie z (4.3)):

Dla σ_2: Dla σ_5:

$x_4 \geq x_2 + t_2$, $x_7 \geq x_3 + t_3$

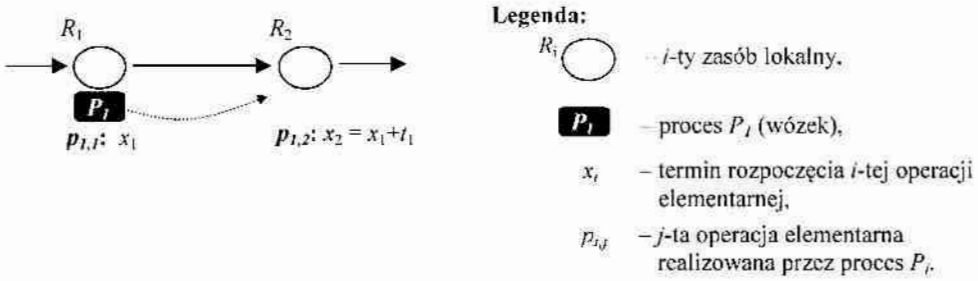
Przyjęto, że na zasobach R_2 i R_5 proces P_1 jest realizowany przed procesem P_2. Więc w przypadku zasobu R_2 operacja $p_{2,1}$ (procesu P_2) może być realizowana po zakończeniu operacji $p_{1,2}$ (procesu P_1), stąd ograniczenie w postaci: $x_4 \geq x_2 + t_2$. Analogicznie dla zasobu R_5 operacja $p_{2,4}$ (procesu P_2) może być realizowana po zakończeniu operacji $p_{1,3}$ (procesu P_1), stąd ograniczenie w postaci: $x_7 \geq x_3 + t_3$.

■

Obsługa procesu przez zasoby lokalne. Przyjęte założenia mówiące o wzajemnym wykluczaniu się zasobów i oczekiwaniu procesu na zwolnienie zasobu, do którego proces żąda dostępu (założenia 3 i 6), determinują zachowanie się procesów na zasobach lokalnych. Podobnie jak poprzednio, założenia te są reprezentowane w postaci odpowiednich ograniczeń narzucanych na zmienne modelu SWPC. Na rysunku 4.7 przedstawiono przykład obsługi procesu P_1 (wózek) przez zasoby lokalne R_1 i R_2. Do procesu P_1 przydzielone są dwie operacje $p_{1,j}$, pierwsza to operacja realizowana na zasobie, na którym wózek aktualnie przebywa, druga to operacja, którą chce wykonać na zasobie kolejnym (do którego żąda dostępu). Wózkowi P_1 przyporządkowane są operacje $p_{1,1}$: x_1 i $p_{1,2}$: x_2;

$p_{1,1}$: x_1 – operacja P_1 na zasobie R_1, która rozpoczyna się w terminie określonym przez x_1,

$p_{1,2}$: x_2 – operacja P_1 na zasobie R_2, która rozpoczyna się w terminie określonym przez x_2.



Rys. 4.7. Obsługa procesu przez zasoby lokalne

Proces P_1 rozpoczyna realizację operacji $p_{1,1}$ na zasobie R_1 w chwili x_1. Po zakończeniu operacji $p_{1,1}$ proces P_1 natychmiast rozpoczyna operacje $p_{1,2}$, na zasobie R_2. Termin rozpoczęcia operacji $p_{1,2}$ (x_2) równy jest czasowi zakończenia operacji $p_{1,1}$ (x_1+t_1). Stąd, dla rozważanego przykładu, termin rozpoczęcia operacji na żądanym zasobie lokalnym można sformułować w postaci następującego wyrażenia:

$$x_2 = x_1 + t_1.$$

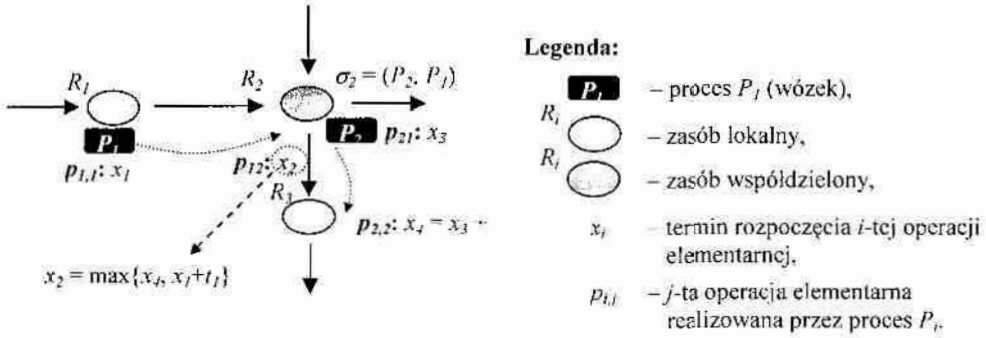
Ogólnie, ograniczenie opisujące wartość terminu rozpoczęcia x_i operacji przez j-ty proces na żądanym zasobie lokalnym ma postać:

$$x_i = x_{i-1} + t_{i-1}, \tag{4.4}$$

gdzie: x_i, x_{i-1} – terminy rozpoczęcia operacji realizowanych przez j-ty proces na sąsiednich zasobach lokalnych.

Obsługa procesu przez zasoby współdzielone. Podobnie jak w przypadku zasobów lokalnych, założenia 3, 5, 6, determinują zachowanie procesów na zasobach współdzielonych. Zachowanie to wyrażane jest w postaci ograniczeń narzucanych na zmienne x modelu SWPC. Na rysunku 4.8 przedstawiono przykład obsługi procesów P_1 i P_2 przez zasób współdzielony R_2. Proces P_1 wykonując operację $p_{1,1}$ na zasobie R_1 żąda dostępu do zasobu R_2 (operacja $p_{1,2}$). Operacja $p_{1,2}$ nie może zostać rozpoczęta dopóki nie zostanie zakończona operacja $p_{1,1}$ i dopóki proces P_2 nie zwolni zasobu R_2, czyli nie rozpocznie operacji $p_{2,2}$ na zasobie R_3. Termin rozpoczęcia operacji $p_{1,2}$ (x_2) jest więc równy terminowi rozpoczęcia operacji $p_{2,2}$ (x_4) lub zakończenia operacji $p_{1,2}$ ($x_1 + t_1$).

Tego typu sytuacja występuje jedynie wtedy gdy reguła obsługi procesów na zasobie R_2 określa pierwszeństwo obsługi procesu P_2.



Rys. 4.8. Obsługa procesów przez zasoby współdzielone

Przedstawioną zasadę obsługi procesu, dla rozważanego przykładu można sformułować w postaci:

$$x_2 = \max\{x_4, x_1+t_1\}.$$

Wykorzystanie operatora max pozwala na określenie terminu rozpoczęcia operacji $p_{1,2}$ tak aby ta operacja rozpoczęła się po zakończeniu operacji $p_{1,1}$, gdy zasób R_2 będzie wolny (P_2 rozpocznie pracę na R_3 – operacja $p_{2,2}$).

Zgodnie z powyższą intuicją, ogólne ograniczenie opisujące wartość terminu rozpoczęcia x_i procesu P_l na zasobie współdzielonym R_v ma postać:

$$x_i = \max\{x_j, x_{i-1}+t_{i-1}\}, \tag{4.5}$$

gdzie: x_i – termin rozpoczęcia operacji przez proces P_l na zasobie współdzielonym R_v,

x_{i-1} – termin rozpoczęcia operacji poprzedzającej operację reprezentowaną przez termin x_i,

x_j – termin rozpoczęcia operacji przez proces P_r (proces, który jest realizowany na R_v przed P_l) na zasobie R_{v+1} (zasób występujący jako następny po R_v w marszrucie procesu P_r).

Termin rozpoczęcia operacji x_i przez proces P_l na zasobie współdzielonym R_v jest równy terminowi ukończenia, przez proces P_l, operacji wcześniejszej x_{i-1} lub terminowi zwolnienia zasobu R_v przez proces P_r pracujący na zasobie R_v przed procesem P_l.

Podsumowując, przedstawione powyżej cztery ograniczenia (4.2), (4.3), (4.4), (4.5), opisujące wzajemne związki między elementami sekwencji x, S_0, Θ, stanowią ogólny opis rozważanej klasy systemów transportowych w kontekście modelu SWPC. W oparciu o nie, możliwa jest budowa postaci faktów F_a stanowiących część reprezentacji bazy wiedzy KB.

Do opisu systemu można wykorzystać dowolną postać faktów F_a pod warunkiem, że przyjęta postać gwarantuje spełnienie powyżej zdefiniowanych ograniczeń, tzn. bez względu na czas, miejsce i rodzaj realizowanego procesu zawsze spełnione będą ograniczenia (4.2), (4.3), (4.4), (4.5). Inaczej mówiąc wymagane jest by ogólne zasady charakteryzujące obiekt reprezentowane były w postaci faktów F_a, które zagwarantują spełnienie tej wiedzy w kontekście wszystkich zmiennych decyzyjnych (w tym przypadku elementów sekwencji x, S_0, Θ).

Należy podkreślić, że spełnienie ograniczeń (4.2), (4.3), (4.4), (4.5), gwarantuje dodatkowo brak wystąpienia kolizji w trakcie pracy procesów [59], [92]. Sformułowane w rozdziale ograniczenia stanowią efekt realizacji pierwszego z przedstawionych w poprzednim punkcie zagadnienia obejmującego proces weryfikacji bazy wiedzy (zagadnienie dotyczące sformułowania założeń klasy systemu w postaci relacji zmiennych modelu SWPC). W dalszej kolejności wymagane jest opisanie stanu blokady w postaci związków (relacji) między zmiennymi modelu SWPC.

4.2.2. Równanie stanu

Rozważany jest uproszczony przypadek gdy: $F(\Theta, S_0, x, P_D) = F_a(\Theta, S_0, x, P_D)$ – czyli zbiór faktów zawiera tylko ogólne fakty odpowiadające ograniczeniom rozważanej klasy systemów. Brak jest natomiast jakichkolwiek dodatkowych faktów $F_b(\Theta, S_0, x, P_D)$ wynikających na przykład z indywidualnych właściwości rozważanego systemu transportowego, takich jak priorytetowanie obsługi określonych wózków, dostępność poszczególnych maszyn, dedykowane zadania pracy poszczególnych wózków, itp.

Realizację procesów w systemie SWPC można opisać poprzez sekwencje stanów $S = (S_0, S_1, \ldots, S_w)$, gdzie: $S_i = (R^{si}_1, R^{si}_2, \ldots, R^{si}_q)$ jest i-tym stanem systemu, elementy którego określają zasoby zajmowane przez procesy $P_1, \ldots, P_q$. W przedstawionej sekwencji elementy $S_0, \ldots, S_w$, reprezentują kolejno stany systemu występujące w oknie czasowym W.

Zwykle w sekwencji stanów S znany jest tylko stan początkowy S_0, pozostałe stany są nieznane. Poszukiwana jest postać sekwencji S. W tym celu wykorzystywana jest graficzna reprezentacja stanów S.

Stan S_i systemu SWPC jest reprezentowany przez graf żądań zasobowych $G^{si} = (N^{si}, B^{si})$, w skład którego wchodzą zbiory N^{si} i B^{si}. W ogólnym przypadku, jest to graf niespójny. Dla danego stanu systemu S_i graf ten określa zasoby, na których realizowane są poszczególne

procesy, terminy rozpoczęcia operacji na zajętych zasobach i żądania dostępu do zasobów współdzielonych. Graf $G^{si} = (N^{si}, B^{si})$ jest charakteryzowany następująco:

- $N^{si} = N^{si}{}_a \cup N^{si}{}_b = \{R^{si}{}_{n,1}, R^{si}{}_{n,2}, \ldots, R^{si}{}_{n,qi}\}$ – jest zbiorem wierzchołków reprezentujących zasoby R, na których w danym stanie S_i realizowane są operacje (zbiór $N^{si}{}_a$) oraz zasoby, do których procesy żądają dostępu po ukończeniu bieżącej operacji (zbiór $N^{si}{}_b$). Każdemu wierzchołkowi jest przyporządkowana współrzędna (etykieta) wierzchołka:

$$\psi(R^{si}{}_{n,i}) = \begin{cases} P_j & \text{– gdy w danym oknie czasowym } W, \text{ w kolejnych stanach systemu, na } R_{n,i} \text{ operacje są realizowane} \\ \Lambda & \text{– gdy w danym oknie czasowym } W, \text{ w kolejnych stanach systemu, na } R^{si}{}_{n,i} \text{ operacje nie są realizowane} \end{cases}$$

  gdzie: P_j – proces, który w kolejnych stanach, jako pierwszy otrzymuje dostęp do zasobu $R^{si}{}_{n,i}$,

  Λ – symbol oznaczający, że w oknie czasowym W na zasobie $R^{si}{}_{n,i}$ operacje nie będą już realizowane.

- $B^{si} = \{B^{si}{}_{a,s}, B^{si}{}_{k,l}, \ldots, B^{si}{}_{y,z}\}$ – jest zbiorem łuków, gdzie: $\|B^{si}\| = q$, $B^{si}{}_{k,l} = (R_i, R_j)$ – łuk łączący wierzchołek R_k z wierzchołkiem R_l, gdzie: $R_k, R_l \in N^{si}$. Łuk $B^{si}{}_{k,l}$ reprezentuje proces realizujący w stanie S_i operację na R_k, po zakończeniu której żąda dostępu do zasobu R_l.
- Każdemu łukowi przyporządkowana jest współrzędna (etykieta) łuku $\varphi(B^{si}{}_{k,l}) = {}^{p}x_v{}^{si}$, gdzie: ${}^{p}x_v{}^{si}$ – współrzędna określająca termin rozpoczęcia operacji p-tego procesu na zasobie R_k.
- Gdy dla łuków $B^{s(i+1)}{}_{k,l}$, $B^{si}{}_{a,s}$ spełnione jest $R_k \neq R_a$ i $R_l \neq R_s$ to współrzędne łuku spełniają nierówność:

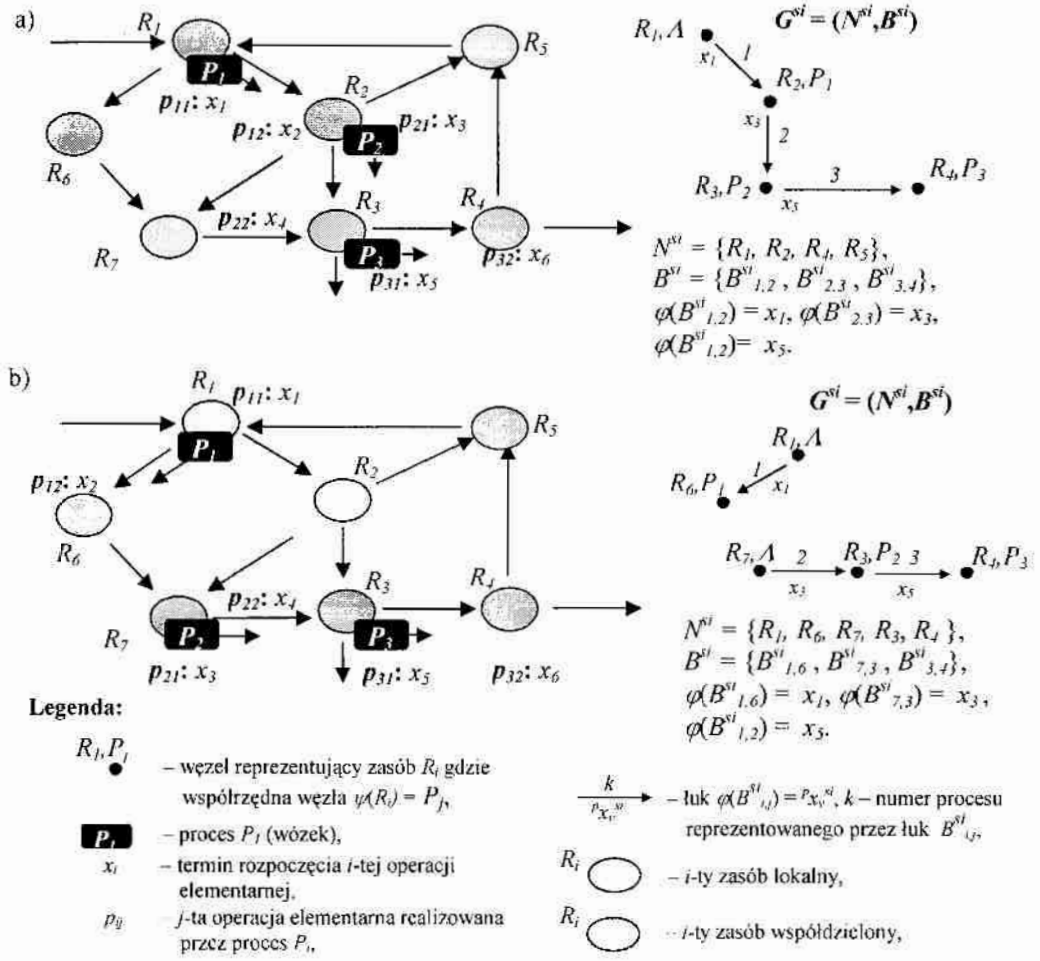
$$ {}^{p}x_v{}^{s(i+1)} \geq {}^{p}x_v{}^{si} + {}^{p}t_v{}^{si}, \tag{4.6}$$

  gdzie: ${}^{p}x_v{}^{si}$ – termin reazlizacji operacji realizowanej przez p-ty proces przed operacją opisaną terminem ${}^{p}x_v{}^{s(i+1)}$,

  ${}^{p}t_v{}^{si}$ – czas trwania operacji reprezentowanej przez ${}^{p}x_v{}^{si}$.

  Zależność (4.6) jest konsekwencją przyjętego ograniczenia (4.2) dotyczącego kolejności realizacji operacji. Termin rozpoczęcia operacji ${}^{p}x_v{}^{s(i+1)}$ w stanie S_{i+1} powinien być większy lub równy od terminu ukończenia operacji $({}^{p}x_v{}^{si} + {}^{p}t_v{}^{si})$ w stanie go poprzedzającym S_i.

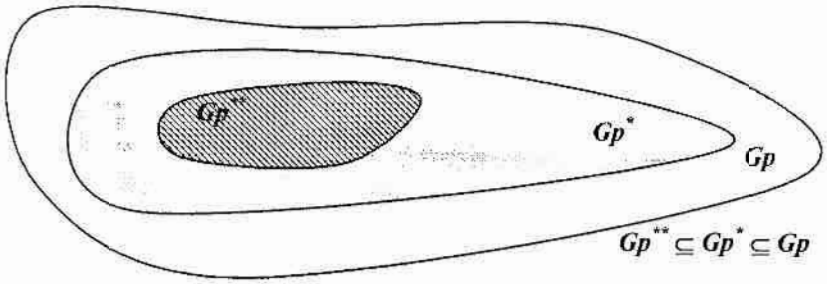
Przykładowe grafy żądań zasobowych reprezentujące określone stany sytemu zostały przedstawione na rysunku 4.9.



Rys. 4.9. Przykłady reprezentacji określonych stanów w postaci grafów: a) stan systemu reprezentowany przez graf spójny, b) stan systemu reprezentowany przez graf niespójny

Zachowanie systemu SWPC jest opisane przez sekwencję grafów żądań zasobowych $G = (G^{s0}, G^{s1}, G^{s2}, \ldots, G^{sw})$ scharakteryzowaną następująco:

- Sekwencja G reprezentuje, w postaci grafów żądań zasobowych G^{si}, wszystkie stany S_i jakie występują w systemie w oknie czasowym W. Grafy G^{si}, występujące w oknie czasowym W, są uporządkowane od grafu początkowego G^{s0} po ostatni graf G^{sw}.
- Każdy graf żądań zasobowych G^{si} posiada tę samą liczbę łuków B^{si} równą liczbie procesów q realizowanych w systemie: $||B^{si}|| = q$. Grafy mogą posiadać różną liczbę wierzchołków N^{si}.
- W rozważanej klasie systemów liczba stanów jest ograniczona przez liczbę operacji elementarnych n realizowanych w oknie W: $||G|| \leq n$
- Zbiór grafów Gp (zbiór wszystkich postaci grafów G^{si}) opisują parametry S_0 i Θ, a tym samym układy nierówności (4.2), (4.3). Parametry S_0 i Θ wpływają na kolejność realizacji procesów w systemie, a tym samym, poprzez układy (4.2) i (4.3), ograniczają potencjalne wartości harmonogramu x. Grafy G^{si} reprezentujące stany systemu SWPC spełniają warunek narzucony na współrzędne łuków (4.6). Zbiór grafów Gp jest zatem ograniczony do zbioru grafów Gp^{**}, zawierającego tylko te grafy, które spełniają układy (4.2), (4.3), oraz własność (4.6) (rysunek 4.10).



Legenda:

Gp – zbiór wszystkich grafów G^{si},

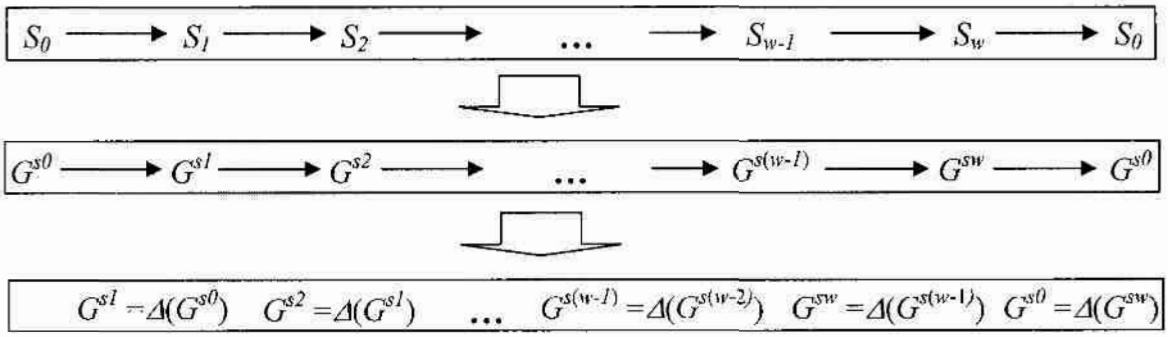
Gp^{*} – zbiór grafów G^{si} spełniających własność (4.6),

Gp^{**} – zbiór grafów G^{si} spełniających własność (4.6) i układy (4.2), (4.3).

Rys. 4.10. Zbiór grafów Gp

W zbiorze grafów Gp^{**} poszukiwane są sekwencje grafów G, które reprezentują bezblokadowe stany systemu.

Na rysunku 4.11 przedstawiono reprezentację sekwencji stanów S w postaci sekwencji grafów żądań zasobowych G. W systemach SWPC procesy realizują swoje operacje cyklicznie, zatem osiągane kolejno stany systemu muszą doprowadzić do ponownego pojawienia się stanu początkowego S_0. Naturalnym jest więc stwierdzenie, że warunkiem poprawnej pracy systemu jest istnienie łańcucha sekwencji stanów, który rozpoczyna się i kończy tym samym stanem S_i (rysunek 4.11). Łańcuch sekwencji stanów powinien gwarantować realizację procesów zgodnie z przydzielonymi im marszrutami.



Legenda:

S_i – i-ty stan systemu G^i – i-ty graf żądań zasobowych

Rys. 4.11. Reprezentacja sekwencji stanów S w postaci sekwencji grafów G

W celu wyznaczenia łańcucha stanów S wykorzystuje się reprezentację w postaci sekwencji grafów żądań zasobowych G. W odróżnieniu od stanów S_i, grafy żądań zasobowych G^{si} stanowią opis stanowy (współrzędne wierzchołków) oraz opis czasowy (współrzędne łuków) realizacji procesów. Inaczej mówiąc, graf G^{si} reprezentuje określony stan systemu S_i wraz z elementami harmonogramu x odpowiadającymi operacjom realizowanym w danym stanie. Wyznaczenie łańcucha stanów S prowadzi do wyznaczenia analogicznego łańcucha grafów żądań zasobowych G (rysunek 4.11).

Sprowadza się to do wyznaczania kolejno grafów G^{si}, aż do ponownego otrzymania grafu G^{s0}. Graf $G^{s(i+1)}$ jest otrzymywany z poprzedzającego go grafu G^{si}. Graf $G^{s(i+1)}$ jest konsekwencją spełnienia przez graf G^{si} żądań zasobowych:

$$G^{s(i+1)} = \Delta(G^{si}), \tag{4.7}$$

gdzie: $\Delta(G^{si})$ – operator spełnienia żądań zasobowych definiowany następująco:

$\Delta(G^a) = G^b$

gdy:

$G^a = (N^a, B^a)$, $G^b = (N^b, B^b)$, $G^b, G^a \in Gp$, $G^b \in Gp^{**}$, $N^b = N^b{}_a \cup N^b{}_b$,

$N^b{}_a \subset N^a$, $N^b{}_a$ – zbiór różnoelemnentowy, współrzędne łuków B^b spełniają ograniczenia spełniają (4.4) i (4.5).

gdzie: $N^b{}_a$ zbiór zasobów R, na których realizowane są operacje w stanie b (współrzędne początkowe łuków B^b).

Efektem spełnienia żądań zasobowych $\Delta(G^{si})$ jest graf $G^{s(i-1)}$ opisany zbiorem łuków $B^{s(i-1)}$ oraz zbiorem wierzchołków $N^{s(i+1)}$. Graf $G^{s(i-1)}$ jest jednym z gafów zbioru $G_\Delta{}^{s(i+1)}$: $G^{s(i+1)} \in G_\Delta{}^{s(i+1)}$, gdzie: $G_\Delta{}^{s(i+1)} \subseteq Gp^{**}$ – jest zbiorem potencjalnych postaci grafu $G^{s(i+1)}$, $G_\Delta{}^{s(i+1)} = \{G_{\Delta 1}{}^{s(i+1)}, G_{\Delta 2}{}^{s(i-1)}, \ldots, G_{\Delta z}{}^{s(i+1)}\}$. Każdy graf zbioru $G_\Delta{}^{s(i+1)}$ opisany jest zbiorem $N_{\Delta i}{}^{s(i+1)}$ charakteryzującym się tym, że podzbiór $N_{\Delta i}{}^{s(i+1)}{}_a$ (zbiór zasobów R, na których realizowane są operacje w stanie $S_{(i+1)}$) jest podzbiorem wierzchołków grafu G^{si}: $N^{s(i-1)}{}_a \subset N^{si}$. Graf G^{si} również należy do zbioru $G_\Delta{}^{s(i+1)}$, co oznacza że w szczególnym przypadku $G^{s(i+1)}$ jest grafem G^{si}. Każdy graf zbioru $G_\Delta{}^{s(i+1)}$ opisany jest ponadto zbiorem łuków $B_{\Delta i}{}^{s(i+1)}$, łuki reprezentują procesy realizujące operacje na zasobach określonych w zbiorze $N_{\Delta i}{}^{s(i+1)}$. Graf $G^{s(i+1)}$ jest grafem, którego wartości współrzędnych łuków $B^{s(i+1)}$ (odpowiadających elementom sekwencji x), spełniają ograniczenia żądań zasobowych (4.4) i (4.5). Ograniczenia te prowadzą do postaci równań stanu, otrzymywanych z zależności (4.8), umożliwiających wyznaczenie współrzędnych łuków grafu $G^{s(i+1)}$ a tym samym określenie jego postaci.

W przypadku gdy G^{si} nie spełnia żądań zasobowych, to wszystkie wartości współrzędnych (zgodnie z (4.8)) wynoszą: ${}^p x_v{}^{si}(x)$, wówczas to graf $G^{s(i+1)}$ ma taką samą postać co G^{si}. W takim przypadku stan systemu nie zmienia się – system osiąga stan blokady.

Równania stanu dla q procesów wyznaczane są z następującego wyrażenia:

$$
{}^p x_v{}^{s(i-1)}(x) = \begin{cases} {}^p x_v{}^{si} + {}^p t_v{}^{si} & \text{gdy } R_{p,si} \text{ jest zasobem lokalnym i } p\text{-ty proces jest reprezentowany przez łuk należący do podgrafu } {}^z G^{si}, \text{ lub gdy } R_{p,si} \text{ jest zasobem współdzielonym i } p\text{-ty proces jest reprezentowany przez łuk należący do podgrafu } {}^z G^{si} \text{ i } R_{p,si} \text{ nie posiada łuku wychodzącego,} \\ \max\{ {}^p x_v{}^{si} + {}^p t_v{}^{si}, {}^{pe} x_v{}^{s(f)} \} & \text{gdy } R_{p,si} \text{ jest zasobem współdzielonym i } p\text{-ty proces jest reprezentowany przez łuk należący do podgrafu } {}^z G^{si} \text{ i } R_{p,si} \text{ posiada łuk wychodzący,} \\ {}^p x_v{}^{si}(x) & \text{gdy } p\text{-ty proces jest reprezentowany przez łuk nie należący do podgrafu } {}^z G^{si}. \end{cases} \qquad (4.8)
$$

dla $p = 1, \ldots, q$

gdzie: ${}^{p}x_v^{s(i+1)}(x)$ – współrzędna łuku, grafu $G^{s(i+1)}$, reprezentującego p-ty proces w stanie S_{i+1},

$R_{p,si}$ – zasób, na którym realizowana jest operacja o terminie rozpoczęcia ${}^{p}x_v^{si}$,

${}^{p}x_v^{si}$ – współrzędna luku reprezentującego proces p w grafie G^{si},

${}^{p}t_v^{si}$ – czas trwania operacji reprezentowanej przez termin ${}^{p}x_v^{si}$,

${}^{pe}x_v^{s(f)}$ – współrzędna łuku opisującego proces P_e w grafie reprezentującym stan S_f (wstępujący przed lub po stanie S_i). Współrzędna jest terminem rozpoczęcia operacji procesu P_e (występującego na zasobie $R_{p,si}$ przed procesem P_p) na zasobie występującym w marszrucie procesu P_e po zasobie $R_{p,si}$.

Proces p reprezentowany jest przez łuk należący do podgrafu ${}^{z}G^{si}$ jeżeli współrzędna ${}^{p}x_v^{si}$ tego łuku należy do zbioru współrzędnych podgrafu ${}^{z}G^{si}$: ${}^{p}x_v^{si} \in {}^{z}x^{si}$, gdzie:

${}^{z}x^{si} = \{{}^{z}x^{si}_1, {}^{z}x^{si}_2, \ldots, {}^{z}x^{si}_{nz}\}$ – zbiór współrzędnych wszystkich łuków wchodzących w skład podgrafu ${}^{z}G^{si}$, do którego należy łuk procesu P_p (numeracja elementów zbioru ${}^{z}x^{si}$ jest zgodna ze zwrotem łuków w grafie ${}^{z}G^{si}$),

$nz = |{}^{z}B|$ - liczba współrzędnych x w podgrafie ${}^{z}G^{si}$.

W wyrażeniu (4.8) podgraf ${}^{z}G^{si}$ oznacza z-ty spójny podgraf grafu G^{si} opisany zbiorami węzłów ${}^{z}N$ i łuków ${}^{z}B$. Wierzchołki i łuki spełniają następujące właściwości podgrafów ${}^{z}G^{si}$:

- Dla każdego wierzchołka ${}^{z}R_{n,j} \in {}^{z}N$ jeżeli istnieją łuki skierowane do tego wierzchołka to co najmniej dla jednego łuku ${}^{z}B_{i,j} = (R_{n,i}, R_{n,j})$ spełniona jest zależność:

$$\psi(R_{n,j}) = pa\,, \tag{4.9}$$

gdzie: pa – oznacza proces reprezentowany przez łuk ${}^{z}B_{i,j}$ skierowany do ${}^{z}R_{n,j}$.

- Dla każdego wierzchołka ${}^{z}R_{n,i} \in {}^{z}N$ i skierowanego od tego wierzchołka łuku ${}^{z}B_{i,j} = (R_{n,i}, R_{n,j})$ spełniona jest zależność:

$$\psi(R_{n,j}) = pr\,, \tag{4.10}$$

gdzie: pr – oznacza proces reprezentowany przez łuk ${}^{z}B_{i,j}$ skierowany od ${}^{z}R_{n,i}$.

W ogólności graf G^{si}, może nie zawierać podgrafu ${}^{z}G^{si}$. Istnienie podgrafu ${}^{z}G^{si}$ jest uzależnione od marszrut procesów P_{pr} określających wartości współrzędnych $\psi(R_{n,i})$ przyporządkowanych wierzchołkom. Maksymalna liczba podgrafów jest ograniczona liczbą procesów realizowanych w systemie $z \le q$ (z – liczba podgrafów, q – liczba realizowanych procesów).

Przykład 4.3. Postacie podgrafów ${}^{z}G^{si}$

Celem przykładu jest ilustracja, dla wybranych grafów G^{si}, postaci podgrafów ${}^{z}G^{si}$ oraz określenie dla każdego z nich zbiorów ${}^{z}x^{si}$.

Rysunek 4.12 przedstawia przykładowe grafy G^{s1}, G^{s2}, G^{s3}.

W przedstawionych grafach kolorem czerwonym oznaczono podgrafy ${}^{z}G^{si}$:

- Dla grafu G^{s1} określono postać podgrafu ${}^{1}G^{s1}$ (przedstawiony rysunek 4.12 a), graf G^{s1}, kolor czerwony). Wszystkie węzły i łuki G^{s1} spełniają założenia (4.9), (4.10),

podgrafu ${}^{z}G^{s1}$. Stąd, zbiór ${}^{1}x^{s1}$ terminów operacji występujących w podgrafie ${}^{1}G^{s1}$ ma postać:

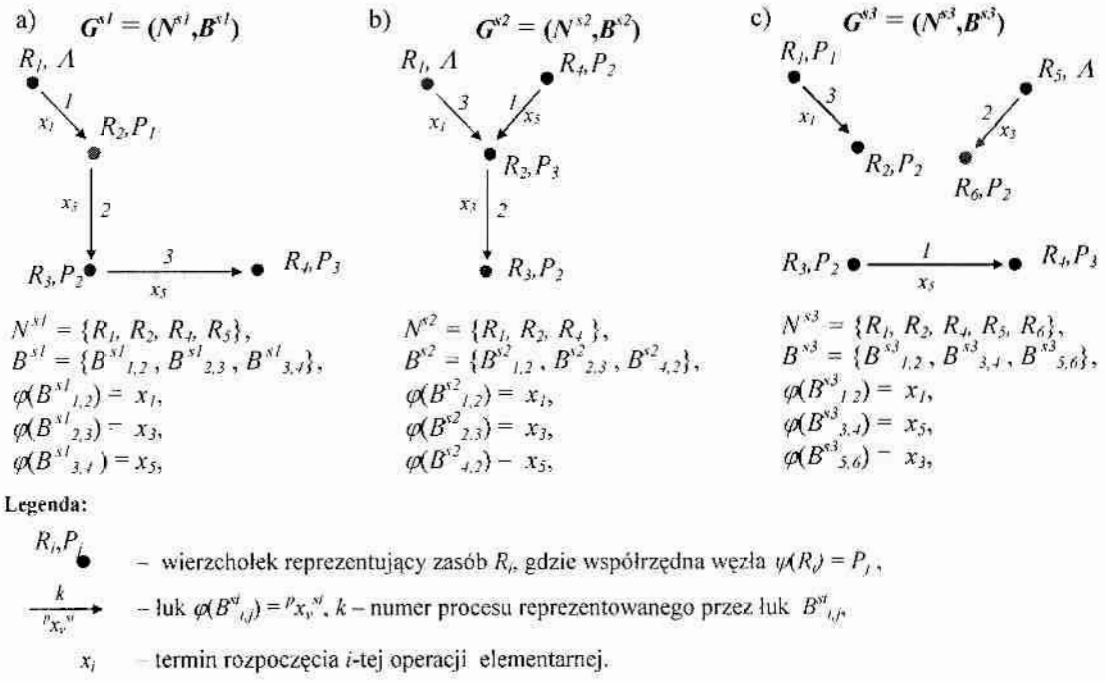
$${}^{1}x^{s1} = \{x_1, x_3, x_5\}.$$

- Dla grafu G^{s2} wyznaczono podgraf ${}^{1}G^{s2}$ (przedstawiony na rysunku 4.12 b), graf G^{s1}, kolor czerwony). Podgraf ${}^{1}G^{s2}$opisany jest zbiorami ${}^{1}N^{s2} = \{R_1, R_2, R_3\}$, ${}^{1}B^{s2} = \{B^{s2}{}_{1,2}, B^{s2}{}_{2,3}\}$. Tylko dla tych zbiorów węzłów i łuków spełnione są założenia (4.9), (4.10), podgrafu ${}^{z}G^{s1}$. Stąd, zbiór ${}^{1}x^{s2}$ terminów operacji wchodzących w skład podgrafu ${}^{1}G^{s1}$ ma postać:

$${}^{1}x^{s2} = \{x_1, x_3\}.$$

- Dla grafu G^{s3} wyznaczono podgraf ${}^{1}G^{s3}$ (przedstawiony na rysunku 4.12 c), graf G^{s1}, kolor czerwony). Podgraf ${}^{1}G^{s3}$ opisany jest zbiorami ${}^{1}N = \{R_5, R_6\}$, ${}^{1}B = \{B_{5,6}\}$. Tylko dla takich zbiorów wierzchołków i luków spełnione są założenia (4.9), (4.10), podgrafu ${}^{z}G^{s1}$. Stąd, zbiór ${}^{1}x^{s1}$ terminów operacji wchodzących w skład podgrafu ${}^{1}G^{st}$ ma postać:

$${}^{1}x^{s3} = \{x_3\}.$$



Rys. 4.12. Przykładowe postacie grafów: a) cały graf spełnia założenia (4.9), (4.10); b) tylko gałęzie 2, 3 spełniają złożenia (4.9), (4.10); c) tylko gałąź 2 spełnia założenia (4.9), (4.10)

W przedstawionych przykładach, dla każdego grafu istnieje tylko jeden podgraf. W ogólności jednak może być ich więcej. Wyznaczone zbiory ${}^{1}x^{s1}$, ${}^{1}x^{s2}$, ${}^{1}x^{s3}$, określają współrzędne łuków reprezentujących procesy, dla których możliwe jest wyznaczenie gałęzi

w kolejnych grafach. Graf G^{si}, dla którego nie istnieje podgraf ${}^{z}G^{si}$ reprezentuje stan, w którym niemożliwe jest wyznaczenie współrzędnych grafu G^{si} co jest równoznaczne ze stanem blokady systemu.

■

Równanie (4.8) formułowane dla każdego (p-tego) procesu realizowanego w systemie, pozwala na wyznaczenie terminu rozpoczęcia operacji w stanie $S_{(i+1)}$ na zasobie, do którego proces żąda dostępu, będąc w stanie S_i.
Dla grafu $G^{si} = (N^{si}, B^{si})$, gdzie:

$$N^{si} = \{R^{si}_{n,1}, R^{si}_{n,2}, \ldots, R^{si}_{n,qi}\}, B^{si} = \{B^{si}_{a,s}, B^{si}_{k,l}, \ldots, B^{si}_{y,z}\}, \varphi(B^{si}_{k,l}) = {}^{p}x_v^{si},$$

graf $G^{s(i+1)} = (N^{s(i+1)}, B^{s(i+1)})$ będący wynikiem operatora $\Delta(G^{si})$ jest opisany zbiorami:

$$N^{s(i-1)} = \{R^{s(i+1)}_{n,1}, R^{s(i-1)}_{n,2}, \ldots, R^{s(i+1)}_{n,q(i+1)}\},$$
$$B^{s(i+1)} = \{B^{s(i+1)}_{a,s}, B^{s(i-1)}_{k,l}, \ldots, B^{s(i+1)}_{y,z}\}, \varphi(B^{s(i+1)}_{k,l}) = {}^{p}x_v^{s(i+1)},$$

gdzie:

p – proces reprezentowany przez łuk $B^{s(i+1)}_{k,l}$.
${}^{p}x_v^{s(i-1)} = {}^{p}x_v^{s(i+1)}(x)$, dla $p = 1,2, \ldots, q$,
${}^{p}x_v^{s(i-1)}(x)$ – współrzędna łuku reprezentującego proces p w stanie S_i. Współrzędna jest wyznaczana z zależności (4.8),
$N^{s(i+1)}$ – zbiór wierzchołków odpowiadających operacjom reprezentowanym przez wyznaczone terminy ${}^{p}x_v^{s(i+1)}$,
$B^{s(i-1)}$ – zbiór łuków odpowiadających operacjom reprezentowanym przez wyznaczone terminy ${}^{p}x_v^{s(i+1)}$.

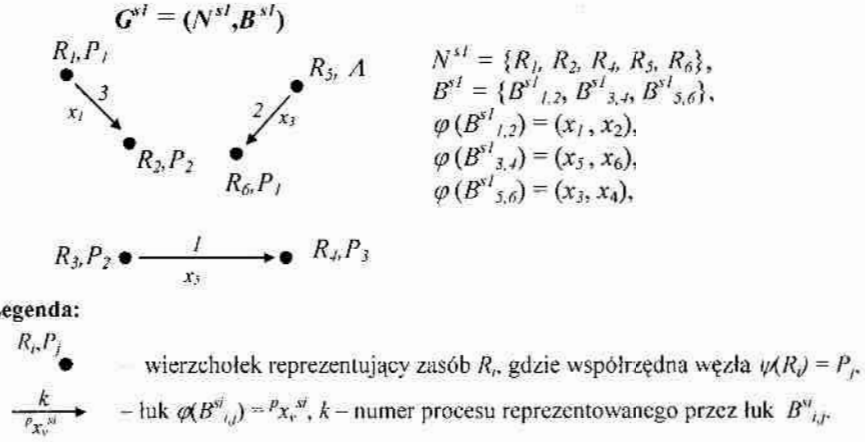
Istota równania (4.8) polega na wykorzystaniu operatora max reprezentującego ograniczenie (4.5) do wyznaczenia łuków wchodzących w skład podgrafu ${}^{z}G^{si}$. Wyznaczenie współrzędnych możliwe jest jedynie dla łuków (procesów) należących do podgrafów ${}^{z}G^{si}$. Współrzędne pozostałych łuków mogą zostać wyznaczone przez równania opisujące kolejne grafy $G^{s(i-k)}$ (graf wstępujący w sekwencji G po grafie G^{si}). Zastosowanie równania (4.8) do sprawdzenia żądań zasobowych może prowadzić do trzech przypadków:

- W grafie G^{si}, istnieje pusty zbiór podgrafów ${}^{z}G^{si}$. Oznacza to że, kolejny graf $G^{s(i+1)}$ ma tą samą postać co G^{si}. Dzieje się tak, gdyż przy braku podgrafu ${}^{z}G^{si}$ wszystkie współrzędne łuków grafu G^{si} przenoszone są do grafu $G^{s(i+1)}$ (zgodnie z (4.8)): ${}^{p}x_v^{s(i+1)}(x) = {}^{p}x_v^{si}(x)$. Wszystkie kolejne stany są takie same, czyli procesy nie mogą być realizowane.
- Cały graf G^{si} spełnia założenia (4.9), (4.10). Oznacza to że, dla współrzędnych wszystkich łuków grafu kolejnego $G^{s(i+1)}$ formułowane są równania stanu (równania otrzymywane z (4.8) w postaci ${}^{p}x_v^{s(i+1)}(x) = {}^{p}x_v^{si} + {}^{p}t_v^{si}$ lub ${}^{p}x_v^{s(i+1)}(x) = \max\{{}^{p}x_v^{si} + {}^{p}t_v^{si}, {}^{pe}x_v^{sf}\}$) umożliwiające wyznaczenie tych współrzędnych. Jeżeli otrzymane równania nie są sprzeczne oznacza to, że przejście systemu do kolejnego stanu $S_{(i+1)}$ odbywa się w wyniku równoległej realizacji operacji wszystkich procesów.

- W grafie G^{si}, istnieje niepusty zbiór podgrafów ${}^{z}G^{si}$. Oznacza to że, w kolejnym grafie $G^{s(i+1)}$ część współrzędnych łuków (należących do ${}^{z}G^{si}$) jest opisana określonym zbiorem równań, pozostałe współrzędne są przenoszone do $G^{s(i+1)}$. Jeśli otrzymane równania są nie sprzeczne to przejście do kolejnego stanu $S_{(i+1)}$ odbywa się w wyniku równoległej realizacji operacji procesów reprezentowanych przez łuki podgrafu ${}^{z}G^{si}$.

Ostanie dwa przypadki opisują sytuacje umożliwiające przejście do stanu kolejnego, pierwszy prowadzi do blokady systemu.

Rozważany jest pierwszy przypadek, gdzie w grafie G^{si} nie można wyznaczyć podgrafu ${}^{z}G^{si}$. W takim przypadku wszystkie łuki B^{si} i wierzchołki N^{si} nie spełniają wyrażeń (4.9), (4.10). Przykładowy graf został przedstawiony na rysunku 4.13.



Rys. 4.13. Przykładowa postać grafu nie zawierającego żadnego podgrafu ${}^{z}G^{si}$

Graf przedstawiony na rysunku 4.13 nie zawiera podgrafu ${}^{z}G^{si}$. Zgodnie z (4.8), w takiej sytuacji nie jest możliwe wyznaczenie wartości żadnej współrzędnej łuków grafu G^{s2} (wszystkie współrzędne przyjmują postać ${}^{\rho}x_v^{s1}(x)$). Jest to równoznaczne z zablokowaniem wszystkich procesów. Okazuje się, że tego typu graf powstaje w przypadku sprzeczności ograniczeń określających kolejność operacji w marszrucie (4.2) i ograniczeń określających kolejność operacji na zasobach współdzielonych (4.3).

W przypadku gdy ograniczenia (4.2) i (4.3) nie są spełnione, graf G^{si} nie zawiera podgrafu ${}^{z}G^{si}$ co oznacza, że system jest zablokowany. Stąd można wyprowadzić następujący wniosek, że każdy graf G^{si}, który należy do zbioru Gp^{**} (zawierający grafy spełniające między innymi ograniczenia (4.2) i (4.3)) zawiera co najmniej jeden podgraf ${}^{z}G^{si}$. Nie oznacza to jednak, że spełnienie (4.2) i (4.3) gwarantuje istnienie rozwiązań bezblokadowych. Równania stanu otrzymane z wyrażenia (4.6) mogą prowadzić do sprzeczności co jest równoznaczne z blokadą systemu. Spełnienie ograniczeń (4.2) i (4.3) powoduje usunięcie tylko pewnej części rozwiązań blokadowych.

Podsumowując, spełnienie żądań zasobowych przez graf G^{si} polega na wyznaczeniu grafu $G^{s(i+1)}$ (w oparciu o zależność (4.8)). Wyznaczenie grafu $G^{s(i+1)}$ polega z kolei na wyznaczeniu współrzędnych łuków $B^{s(i+1)}$, a tym samy wierzchołków $N^{s(i+1)}$. Procesy reprezentowane przez łuki, dla których wyznaczono współrzędne, rozpoczynają operacje na

kolejnych zasobach (wynikających z marszrut P_i). W ten sposób system osiąga nowy stan S_{i-1} reprezentowany przez graf $G^{s(i-1)}$.

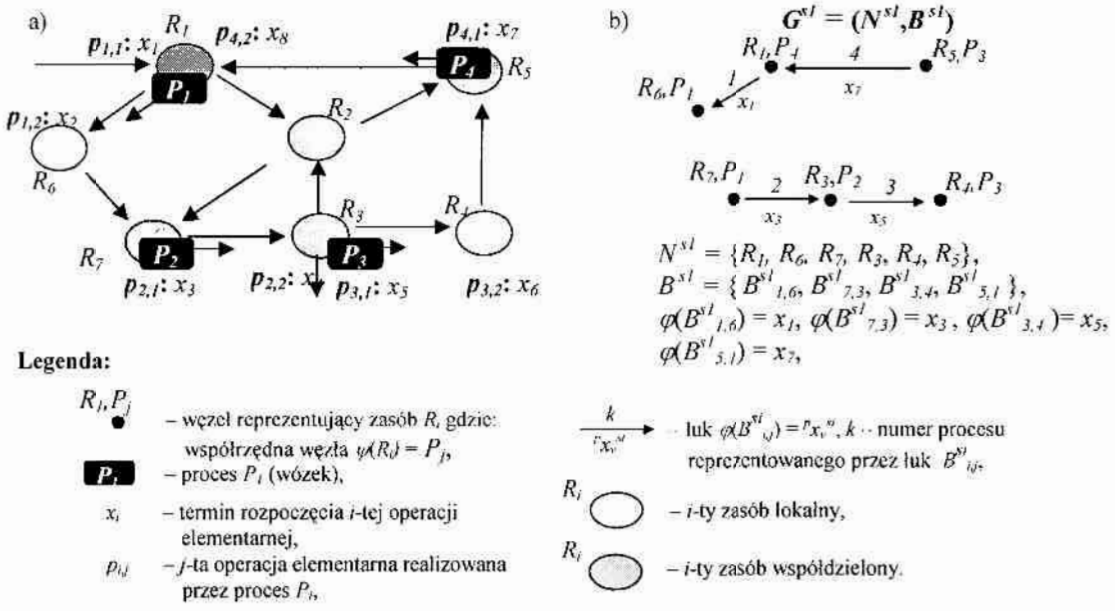
Postępując analogicznie, z grafu $G^{s(i+1)}$ wyznaczany jest graf $G^{s(i+2)}$ i dalej grafy kolejne. Stworzony w ten sposób łańcuch grafów żądań zasobowych odpowiada sekwencji stanów S osiąganych w systemie (rysunek 4.11). Grafy wyznaczane są do momentu otrzymania grafu początkowego S_0.

Przykład 4.4. Idea formułowania równań stanu

Celem przykładu jest ilustracja równań stanu dla stanu systemu z rysunku 4.14 oraz równań stanu dla stanu kolejnego.

Dany jest system z rysunku 4.14. Przyjęte marszruty procesów P, stan początkowy S_0 i sekwencja reguł priorytetowania Θ, implikują graf systemu postaci jak na rysunku 4.14.

W danym stanie G^{s1} znane są wartości parametrów x_1, x_3, x_5, x_7 (współrzędne początkowe łuków). Spełnienie żądań zasobowych dla grafu G^{s1} polega na wyznaczeniu grafu G^{s2}. Poszukiwane są więc wartości x_2, x_4, x_6, x_8 (współrzędne łuków G^{s2}).



Rys. 4.14. Przykład stanu systemu SWPC: a) stan S_1, b) graf G^{s1}

W grafie żądań zasobowych G^{s1} wyróżnia się dwa podgrafy:

$${}^{1}G^{s1} = ({}^{1}N^{s1}, {}^{1}B^{s1}) \quad \text{gdzie: } {}^{1}N^{s1} = \{R_1, R_6, R_5\}, {}^{1}B^{s1} = \{B^{s1}_{5,1}, B^{s1}_{1,6}\},$$
$${}^{2}G^{s1} = ({}^{2}N^{s1}, {}^{2}B^{s1}) \quad \text{gdzie: } {}^{2}N^{s1} = \{R_7, R_3, R_4\}, {}^{2}B^{s1} = \{B^{s1}_{7,3}, B^{s1}_{1,6}\},$$
$$G^{s1} = (N^{s1}, B^{s1}) \quad \text{gdzie: } N^{s1} = {}^{1}N^{s1} \cup {}^{2}N^{s1}, \; B^{s1} = {}^{1}B^{s1} \cup {}^{2}B^{s1}.$$

Współrzędne łuków wchodzących w skład grafów ${}^{1}G^{s1}$, ${}^{2}G^{s1}$, wyrażone są w postaci zbiorów:

$${}^{1}x^{s1} = \{x_7, x_1\},$$
$${}^{2}x^{s1} = \{x_3, x_5\}.$$

Zgodnie z równaniem (4.8) współrzędne łuków grafu G^{s2} mają postać:

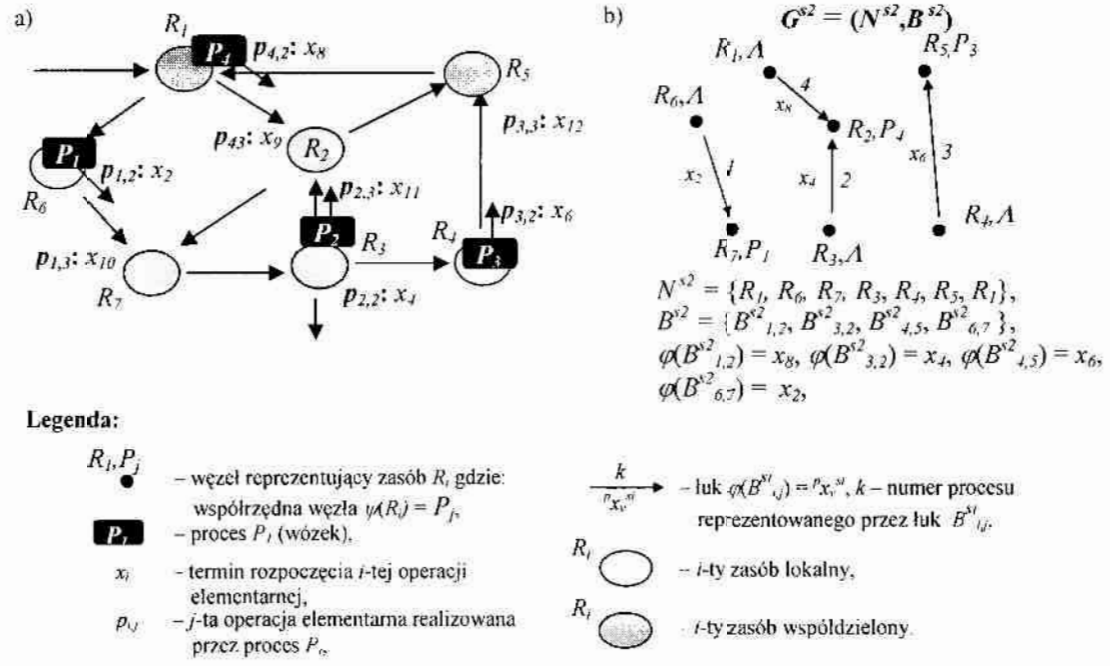
$$\text{Dla } P_1\text{: } x_2 = {}^1x_v^{s2}(x) = x_1 + t_1,$$
$$\text{Dla } P_2\text{: } x_4 = {}^2x_v^{s2}(x) = \max\{x_3 + t_3,\ {}^3x_v^{s2}(x)\},$$
$$\text{Dla } P_3\text{: } x_6 = {}^3x_v^{s2}(x) = x_5 + t_5,$$
$$\text{Dla } P_4\text{: } x_8 = {}^4x_v^{s2}(x) = \max\{x_7 + t_7,\ {}^1x_v^{s2}(x)\}.$$

Otrzymane równania stanu dla G^{s2} mają postać:

$${}^1x_v^{s2}(x) = x_1 + t_1,$$
$${}^2x_v^{s2}(x) = \max\{x_3 + t_3,\ x_5 + t_5\},$$
$${}^3x_v^{s2}(x) = x_5 + t_5,$$
$${}^4x_v^{s2}(x) = \max\{x_7 + t_7,\ x_1 + t_1\}.$$

W oparciu o powyższe równania możliwe jest wyznaczenie poszukiwanych współrzędnych x_2, x_4, x_6, x_8. Wyznaczenie tych współrzędnych pozwala na przejście do następnego grafu G^{s2}. Graf G^{s2} został przedstawiony na rysunku 4.15. Nowe współrzędne poszczególnych węzłów $\psi(R_{n,i})$ grafu G^{s2} wynikają z przyjętych marszrut P realizowanych procesów.

Postępując analogicznie, w oparciu o równania stanu oraz o wyznaczone wartości x_2, x_4, x_6, x_8, można wyznaczyć kolejny graf G^{s3}, a z niego wyznaczać następne.



Rys. 4.15. Przykład stanu systemu SWPC: a) stan S_2, b) graf G^{s2}

■

Przedstawione podejście umożliwia wyznaczanie łańcucha grafów żądań zasobowych (rysunek 4.11) poprzez kolejne wyznaczanie grafów G^{si}. W tym celu wprowadza się pojęcie

stanu osiągalnego. Dany jest zbiór procesów, ich marszruty P, stan początkowy S_0 oraz reguły priorytetowania Θ.

Definicja 4.1

Stanem osiągalnym nazywamy taki stan S_i, któremu odpowiada graf osiągalny $G^{si} \in Gp^{**}$ otrzymany z grafu G^{s0} (reprezentującego stan początkowy S_0) w wyniku skończonej, i-tej liczby spełnień żądań zasobowych:

$$G^{si} = \Delta^{(i)}(\Delta^{(i-1)}(\dots \Delta^{(0)}(G^{s0}) \dots)),$$

gdzie: $\Delta^{(i)}(G^{sj})$ – i-ta realizacja operatora spełnienia żądań zasobowych.

Przedstawiona definicja stanowi podstawę do wyznaczania stanu S_{i+k} (reprezentowanego przez $G^{s(i+k)}$) w oparciu o osiągalny stan przeszły S_i (reprezentowany grafem G^{si}). Wyznaczenie grafu $G^{s(i+k)}$, przy znajomości postaci grafu osiągalnego G^{si}, jest możliwe w wyniku wielokrotnego spełnienia przez graf G^{si} żądań zasobowych:

$$G^{s(i+k)} = \Delta^{(k)}(\Delta^{(k-1)}(\dots \Delta^{(i)}(G^{si}) \dots)),$$

gdzie: $\Delta^{(i)}(G^{sj})$ – i-ta realizacja operatora spełnienia żądań zasobowych.

Wyznaczenie grafu $G^{s(i+k)}$ polega więc na wyznaczeniu współrzędnych łuków i w oparciu o nie, określeniu zbioru wierzchołków $N^{s(i+k)}$ oraz łuków $B^{s(i+k)}$. W tym celu wykorzystuje się następujący lemat:

Lemat 4.1

Dany jest stan osiągalny S_i reprezentowany przez graf żądań zasobowych G^{si}. Dane są marszruty procesów P_i, reguły priorytetowania Θ, grafy $G^{s(i+1)}$, $G^{s(i+2)}$, ... , $G^{s(i+k-1)} \in Gp^{**}$, które spełniają ograniczenia (4.4), (4.5). Stan S_{i+k} reprezentowany jest przez graf żądań zasobowych $G^{s(i+k)} \in Gp^{**}$, gdzie współrzędne łuków opisuje równanie:

$${}^{p}X_v^{s(i+k)}(x) = {}^{ps}x_v^{s(i+k)}({}^{ps}x_v^{s(i+k-1)}({}^{ps}x_v^{s(i+k-2)}(\dots {}^{ps}x_v^{si}(x) \dots))) \text{ dla } p, ps = 1, \dots , q \qquad (4.11)$$

gdzie: ${}^{p}X_v^{s(i+k)}$ – współrzędna x_v łuku B reprezentującego p-ty proces w grafie $G^{s(i+k)}$.

Dowód

G^{si} jest grafem osiągalnym, zatem $G^{si} \in Gp^{**}$. W grafie G^{si} istnieje niepusty zbiór podgrafów ${}^{z}G^{si}$. Kolejny graf $G^{s(i+1)}$ należy do Gp^{**} i spełnia ograniczenia (4.4), (4.5) więc możliwe jest w oparciu o (4.8) wyznaczenie współrzędnych łuków grafu $G^{s(i+1)}$. $G^{s(i+1)}$ jest zatem również grafem osiągalnym. Podobnie dla kolejnych grafów, jeśli należą do Gp^{**} i spełniają (4.4), (4.5) to spełniają żądania zasobowe, czyli istnieje łańcuch łączący graf G^{si} z $G^{s(i+k)}$. Współrzędne łuków grafu $G^{s(i+k)}$ wyznaczane są z zależności (4.8), której argumentami są współrzędne grafu $G^{s(i+k-1)}$ poprzedzającego graf $G^{s(i+k)}$ w łańcuchu grafów. Z kolei do wyznaczenia współrzędnych końcowych $G^{s(i+k-1)}$ wykorzystuje się zależność (4.8) z argumentami współrzędnych grafu $G^{s(i+k-2)}$. W konsekwencji współrzędne $G^{s(i+k)}$ są wyznaczane rekurencyjnie. Stąd słuszna jest zależność (4.11).

c.k.d.

Równania (4.11) nazywane są ogólnymi równaniami stanu. Pozwalają one określić czy istnieje w zbiorze grafów Gp^{**} ścieżka łącząca kilka kolejnych osiągalnych grafów żądań zasobowych G^{si}, $G^{s(i-1)}$, ... , $G^{s(i+k)}$. Równania (4.11) pozwalają wyznaczyć współrzędne łuków grafu $G^{s(i-k)}$ tylko i wyłącznie w oparciu o współrzędne łuków grafu G^{si}. Nie jest wymagana znajomość współrzędnych x grafów pośrednich: $G^{s(i+1)}$, $G^{s(i+2)}$, ... , $G^{s(i+k-1)}$.

Lemat 4.1 prowadzi również do następującego wniosku: Jeśli z G^{si} możliwe jest wyznaczenie $G^{s(i+k)}$, to wszystkie grafy pośrednie G^{sj}, gdzie: $i < j < i+k$, spełniają żądania zasobowe.

Posługując się Lematem 4.1, w szczególnym przypadku, można sformułować równania umożliwiające wyznaczenie ze stanu początkowego S_0 stan ostatni S_w okna czasowego W. Czyli z pierwszego grafu G^{s0} wyznaczyć graf G^{sw}. Ogólne równanie stanu dla systemów opisanych sekwencją żądań zasobowych $G = (G^{s0}, G^{s1}, G^{s2}, \ldots, G^{sw})$ ma postać:

$$ {}^{p}X_v^{sw}(x) = {}^{p}x_v^{sw}({}^{p}x_v^{s(w-1)}({}^{p}x_v^{s(w-2)}(\ldots {}^{p}x_v^{s0}(x) \ldots))), \text{ dla } p = 1, \ldots, q, \qquad (4.12) $$

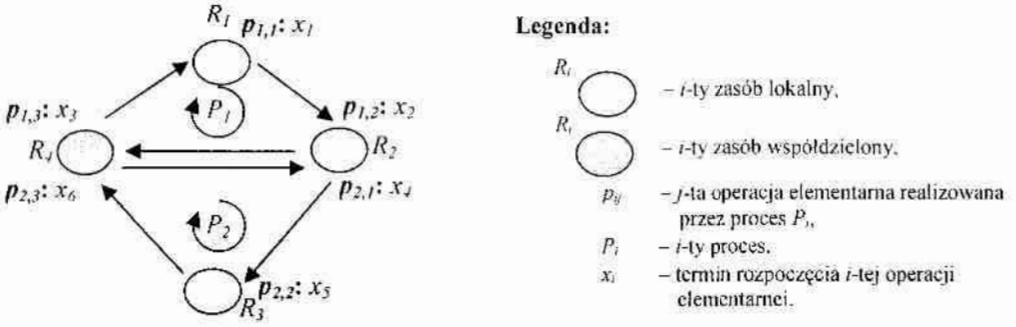
gdzie: w – liczba grafów wchodzących w skład sekwencji żądań zasobowych G,

${}^{p}X_v^{sw}$ – współrzędna ${}^{p}x_v^{sw}$ łuku reprezentującej proces P_p w stanie G^{sw}.

Równania (4.11), (4.12) stanowią opis czasowo-stanowy systemów SWPC. Wynikają one bezpośrednio z przyjętych reguł i zasad opisanych zależnościami (4.2), (4.3), (4.4), (4.5). Opis tego typu stanowi podstawę do wyznaczania warunków wystarczających systemu, tzn. warunków prowadzących do rozwiązań bezblokadowych i bezkolizyjnych.

Przykład 4.5. Ilustracja grafów żądań zasobowych w oknie czasowym *W*

Przykład ma na celu ilustrację grafów żądań zasobowych reprezentujących realizację procesów w oknie czasowym W. Dany jest system z rysunku 4.16.



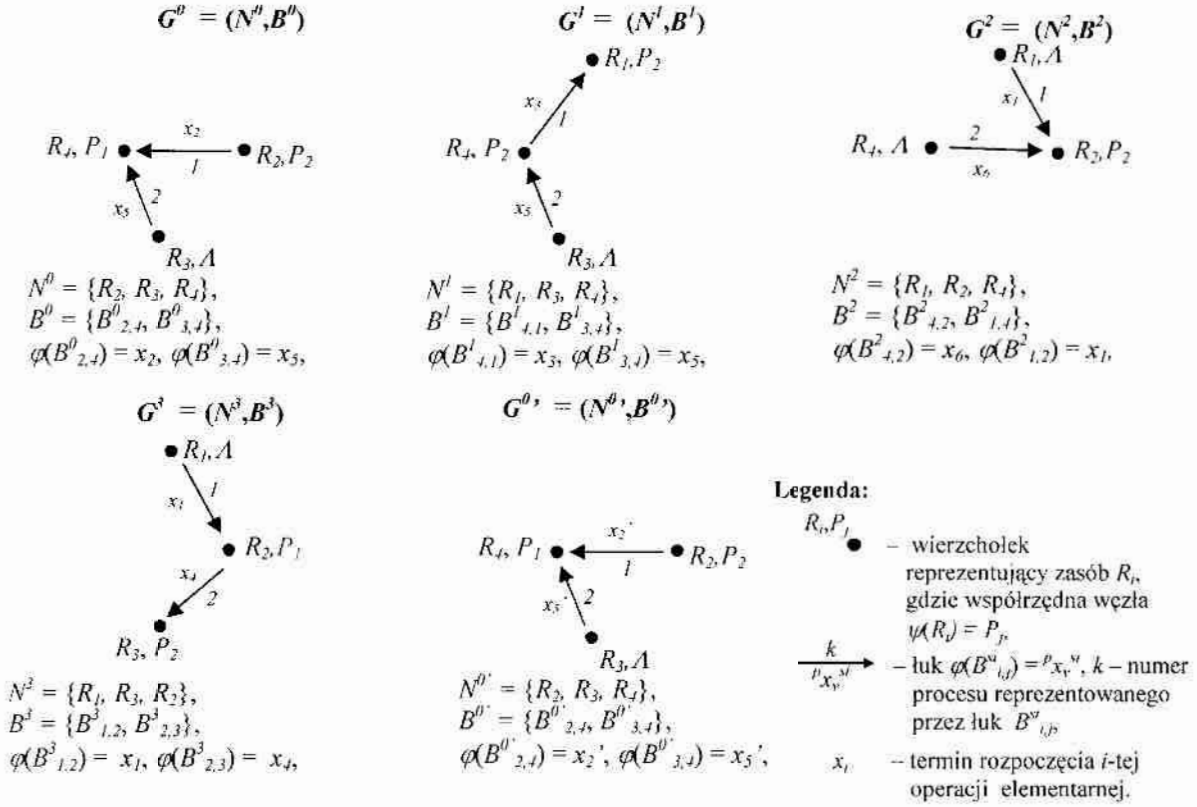
Rys. 4.16. Przykład systemu SWPC

Parametry systemu w oknie czasowym W:

$$P = \{P_1, P_2\}, P_1 = (R_1, R_2, R_4), P_2 = (R_2, R_3, R_4),$$
$$po = (p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{2,3}),$$
$$x = (x_1, x_2, x_3, x_4, x_5, x_6),$$
$$t = (t_1, t_2, t_3, t_4, t_5, t_6).$$

Przyjęty stan początkowy S_0 i reguły priorytetowania Θ mają postać:

$S_0 = (R_2, R_3)$, $\Theta = (\sigma_2, \sigma_4)$, $\sigma_2 = (P_1, P_2)$, $\sigma_4 = (P_1, P_2)$.



Rys. 4.17. Grafy reprezentujące realizację systemu w oknie czasowym W

Na rysunku 4.16 przedstawiono kolejno 5 grafów. Grafy G^0, G^1, G^2, G^3, $G^{0'}$, reprezentują kolejne stany S_0, S_1, S_2, S_3 systemu, w oknie czasowym W oraz stan S_0 w kolejnym oknie. Sekwencja grafów żądań zasobowych dla W ma postać $G = (G^0, G^1, G^2, G^3)$. Zgodnie z (4.8), równania stanu dla grafów G^0, G^1, G^2, G^3, $G^{0'}$, mają postać:

Dla G^0:

$$^1x_v^0(x) = x_2,$$
$$^2x_v^0(x) = x_5.$$

Dla G^1:

$$^1x_v^1(x) = x_3 = x_2 + t_2,$$
$$^2x_v^1(x) = {}^2x_v^0(x).$$

Dla G^2:

$$^1x_v^2(x) = x_1 = x_3 + t_3,$$
$$^2x_v^2(x) = x_6 = \max\{x_5 + t_5, x_3 + t_3\}.$$

Dla G^3:

$$^1x_v^3(x) = {}^1x_v^2(x),$$
$$^2x_v^3(x) = x_4 = x_6 + t_6.$$

Dla $G^{0'}$:

$$^1x_v^{0'}(x) = x_2' = \max\{x_1 + t_1, x_4 + t_4\},$$
$$^2x_v^{0'}(x) = x_5' = x_4 + t_4.$$

Powyższe równania pozwalają na wyznaczenie grafów G^1, G^2, G^3 tworzących łańcuch grafów żądań zasobowych.

Równania stanu, umożliwiające wyznaczenie grafu $G^{0'}$, mają postać (zgodnie z (4.11)):

$$^1X_v^{0'}(x) = x_2' = \max\{(x_2+t_2+t_3+t_1), \max\{(x_5+t_5), (x_2+t_2+t_3+t_6+t_4)\}\},$$
$$^2X_v^{0'}(x) = x_5' = \max\{(x_5+t_5), (x_2-t_2+t_3)\}+t_6+t_4 .$$

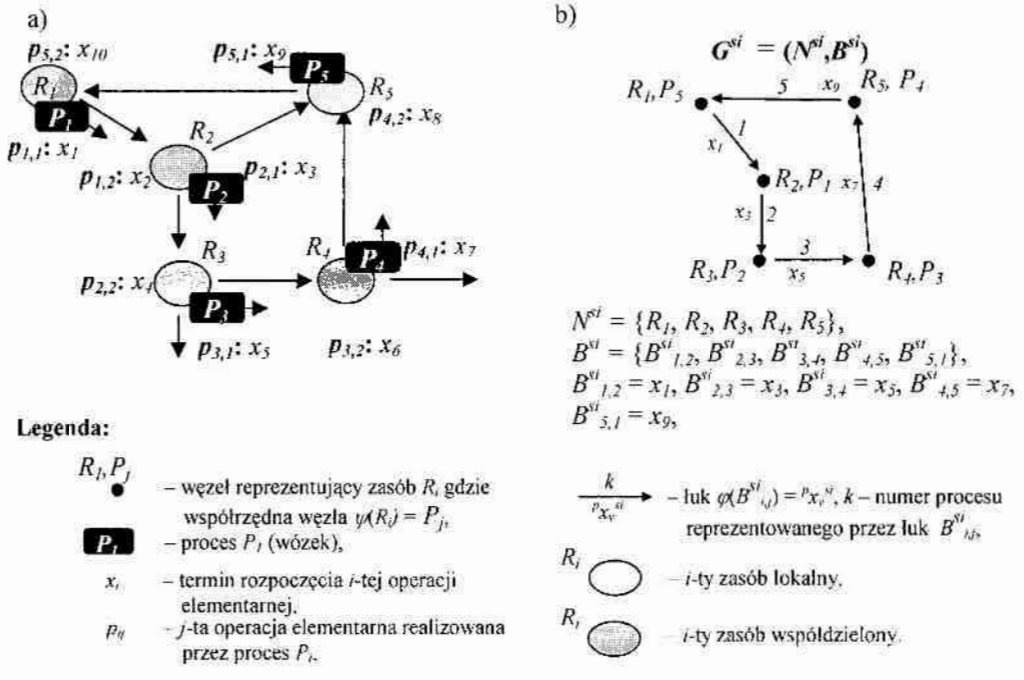
Równania te pozwalają na wyznaczenie współrzędnych grafu $G^{0'}$ tylko w oparciu o współrzędne (x_2, x_5) łuków grafu G^0. Do wyznaczenia grafu G^0 nie są więc potrzebne współrzędne grafów pośrednich.

■

Przedstawione podejście do reprezentowania stanów modelu SWPC w postaci grafów żądań zasobowych, pozwoliło na wyznaczenie związków pomiędzy kolejnymi stanami systemu (a tym samym między zmiennymi modelu SWPC) w postaci ogólnych równań stanu. Równania te pozwalają, w oparciu o stan początkowy S_0, na wyznaczenie kolejnych osiągalnych stanów spełniających ograniczenia (4.2), (4.3), (4.4), (4.5). Opis ten stanowi podstawę do scharakteryzowania stanu blokady niezbędnego do weryfikacji bazy wiedzy. W następnym rozdziale omówiono sposób opisu stanu blokady w oparciu o zaproponowane powyżej podejście.

4.2.3. Blokada w systemie współbieżnych procesów cyklicznych

Na rysunku 4.18 przedstawiono stan systemu SWPC zwany stanem blokady. Stan blokady charakteryzuje się tym, że żądania zasobowe określonych procesów nie mogą zostać spełnione gdyż procesy tworzą cykl żądań zasobowych.



Rys. 4.18. Przykład stanu blokady: a) stan systemu S_i, b) graf żądań zasobowych G^{si}

Definicja 4.1

Blokada jest stanem S_i systemu, któremu odpowiada graf $G^{si} = (N^{si}, B^{si})$ zawierający co najmniej jeden podgraf ${}^{z}G^{si}$ będący cyklem.

Definicja 4.2

Jeżeli podgraf ${}^{z}G^{si}$ jest cyklem i ${}^{z}G^{si} = G^{si}$ to stan S_i jest nazywany stanem blokady całkowitej. Znaczy to, że wszystkie procesy systemu są zablokowane.

Definicja 4.3

Jeżeli ${}^{z}G^{si}$ jest grafem zawierającym cykl i ${}^{z}G^{si}$ jest podgrafem G^{si}, takim że ${}^{z}N^{si} \subset N^{si}$ i ${}^{z}B^{si} \subset B^{si}$, wtedy odpowiadający mu stan jest nazywany stanem blokady częściowej. Znaczy to, że w stanie blokady znajduje się tylko część wszystkich procesów systemu.

Przykład 4.6. Równanie stanu dla blokady

Celem przykładu jest ilustracja procesu budowy równań stanu dla stanu z rysunku 4.18. Rozważmy system w stanie blokady przedstawiony na rysunku 4.18. W danym grafie G^{si} znane są wartości współrzędnych x_1, x_3, x_5, x_7, x_9. Poszukiwane są wartości współrzędnych $x_2, x_4, x_6, x_8, x_{10}$, grafu $G^{s(i+1)}$. Zgodnie z (4.8):

$$\text{dla } P_1\text{: } x_2 = {}^{1}x_v^{s2}(x) = \max\{x_1 + t_1, {}^{2}x_v^{s2}(x)\},$$
$$\text{dla } P_2\text{: } x_4 = {}^{2}x_v^{s2}(x) = \max\{x_3 + t_3, {}^{3}x_v^{s2}(x)\},$$
$$\text{dla } P_3\text{: } x_6 = {}^{3}x_v^{s2}(x) = \max\{x_5 + t_5, {}^{4}x_v^{s2}(x)\},$$
$$\text{dla } P_4\text{: } x_8 = {}^{4}x_v^{s2}(x) = \max\{x_7 + t_7, {}^{5}x_v^{s2}(x)\},$$
$$\text{dla } P_5\text{: } x_{10} = {}^{5}x_v^{s2}(x) = \max\{x_9 + t_9, {}^{1}x_v^{s2}(x)\}.$$

Zgodnie z powyższym, równania stanu dla rozważanego systemu mają postać:

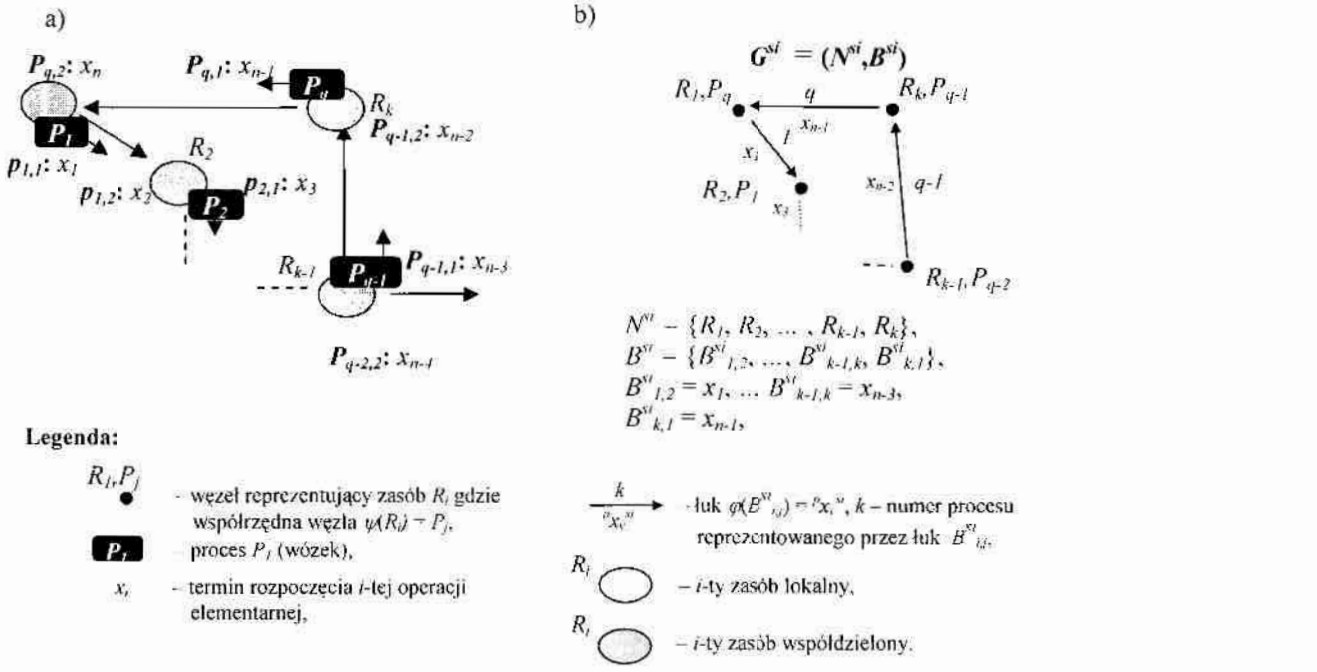
$${}^{1}x_v^{s2}(x) = x_2 = \max\{x_1 + t_1, \max\{x_3 + t_3, \max\{x_5 + t_5, \max\{x_7 + t_7, \max\{x_9 + t_9, x_2\}\}\}\}\},$$
$${}^{2}x_v^{s2}(x) = x_4 = \max\{x_3 + t_3, \max\{x_5 + t_5, \max\{x_7 + t_7, \max\{x_9 + t_9, \max\{x_1 + t_1, x_4\}\}\}\}\},$$
$${}^{3}x_v^{s2}(x) = x_6 = \max\{x_5 + t_5, \max\{x_7 + t_7, \max\{x_9 + t_9, \max\{x_1 + t_1, \max\{x_3 + t_3, x_6\}\}\}\}\},$$
$${}^{4}x_v^{s2}(x) = x_8 = \max\{x_7 + t_7, \max\{x_9 + t_9, \max\{x_1 + t_1, \max\{x_3 + t_3, \max\{x_5 + t_5, x_8\}\}\}\}\},$$
$${}^{5}x_v^{s2}(x) = x_{10} = \max\{x_9 + t_9, \max\{x_1 + t_1, \max\{x_3 + t_3, \max\{x_5 + t_5, \max\{x_7 + t_7, x_{10}\}\}\}\}\}.$$

Rozwiązując przedstawione równania okazuje się, że charakteryzują się one następującymi właściwościami:

- Dla $(x_i < x_1 + t_1) \vee (x_i < x_3 + t_3) \vee (x_i < x_5 + t_5) \vee (x_i < x_7 + t_7) \vee (x_i < x_9 + t_9)$, gdzie: $i = 2, 4, 6, 8, 10$, równania stanu opisujące stan systemu z rysunku 4.18 są sprzeczne.
- Dla $(x_i \geq x_1 + t_1) \wedge (x_i \geq x_3 + t_3) \wedge (x_i \geq x_5 + t_5) \wedge (x_i \geq x_7 + t_7) \wedge (x_i \geq x_9 + t_9)$, gdzie: $i = 2, 4, 6, 8, 10$, równania stanu opisujące stan systemu z rysunku 4.18 są tożsame.

■

Rozważmy ogólny przypadek stanu blokady dla q – procesów (rysunek 4.19).



Rys. 4.19. Przykład blokady dla q – procesów: a) stan systemu S_i, b) graf żądań zasobowych G^{si}

Analogicznie do przykładu 4.6, równania stanu, dla procesów reprezentowanych przez łuki należące do podgrafu ${}^{z}G^{si}$ tworzącego cykl, mają postać:

$$
\begin{aligned}
{}^{p}x_v^{s(i+1)}(x) = \max\{{}^{p}x_v^{si} + {}^{p}t_v^{si}, \max\{{}^{(p-1)}x_v^{si} + {}^{(p-1)}t_v^{si}, \ldots, \max\{{}^{(qz-1)}x_v^{si} + {}^{(qz-1)}t_v^{si}, \max\{{}^{qz}x_v^{si} \\
+ {}^{qz}t_v^{si}, \max\{{}^{1}x_v^{si} + {}^{2}t_v^{si}, \max\{{}^{2}x_v^{si} + {}^{2}t_v^{si}, \ldots, \max\{{}^{(p-1)}x_v^{si} + {}^{(p-1)}t_v^{si}, {}^{p}x_v^{s(i+1)}(x)
\end{aligned}
\qquad (4.13)
$$

$\}\ldots\}\}\}\}$

dla $p = 1, \ldots, q_z$,

gdzie: $q_z = |[{}^{z}B^{si}]|$ – liczba łuków (procesów) wchodzących w skład podgrafu grafu ${}^{z}G^{si}$,

${}^{p}x_v^{s(i+1)}(x)$ – współrzędna łuku grafu $G^{s(i+1)}$ reprezentującego p-ty proces w stanie S_{i+1},

${}^{p}x_v^{si}$ – współrzędna łuku reprezentującego proces p w grafie G^{si},

${}^{z}t^{si}_{j}$ – czas trwania operacji reprezentowanej przez termin rozpoczęcia ${}^{z}x^{si}_{j}$.

Podobnie jak w przykładzie 4.6 okazuje się, że przedstawione równania (4.13) charakteryzują się własnościami:

- Dla ${}^{p}x_v^{s(i+1)} < {}^{pl}x_v^{si} + {}^{pl}t_v^{si}$, $p = 1, \ldots, q_z$, $pl = 1, \ldots, q_z$, równanie (4.13) jest sprzeczne,
- Dla ${}^{p}x_v^{s(i+1)} \geq {}^{pl}x_v^{si} + {}^{pl}t_v^{si}$, $p = 1, \ldots, q_z$, $pl = 1, \ldots, q_z$, równanie (4.13) jest tożsamościowe.

Przedstawione właściwości prowadzą do następującego lematu:

Lemat 4.2

Dany jest graf G^{si} reprezentujący stan osiągalny S_i. Graf G^{si} zawiera podgraf ${}^{z}G^{si}$. Jeżeli podgraf ${}^{z}G^{si}$ jest cyklem, to równanie stanu (4.8) opisujące ${}^{z}G^{si}$ jest tożsamościowe lub sprzeczne.

Dowód

Jeżeli ${}^{z}G^{si}$ jest cyklem, to równanie (4.8) opisujące ${}^{z}G^{si}$ przyjmuje postać równania (4.13). Z własności tego równania wynika, że jest ono tożsamościowe lub sprzeczne. Stąd też równanie (4.8) dla grafów tworzących cykl jest tożsamościowe lub sprzeczne.

c.k.d.

Dla równań stanu o charakterze tożsamościowym, Lemat 4.2 jest również spełniony w kierunku odwrotnym:

Lemat 4.3

Dany jest graf G^{si} reprezentujący stan osiągalny S_i. Graf G^{si} zawiera podgraf ${}^{z}G^{si}$. Jeżeli równanie stanu (4.6) opisujące ${}^{z}G^{si}$ jest tożsamościowe to, podgraf ${}^{z}G^{si}$ jest cyklem.

Dowód

Jeśli S_i jest stanem osiągalnym to graf $G^{si} \in Gp^{**}$, czyli spełnione są ograniczenia kolejnościowe (4.2), (4.3), oraz G^{si} spełnia ograniczenia (4.4), (4.5). Zgodnie z zależnością (4.8), współrzędna ${}^{p}x_v^{si}(x)$ może przyjąć jedną z trzech postaci. Łatwo dostrzec, że tylko postać odpowiadająca zagnieżdżonym operatorom max może mieć charakter tożsamościowy. Odpowiada ona zależności (4.13). Zatem gdy równanie (4.8) jest tożsamościowe przyjmuje ono postać równania postaci (4.13) (opisującego stan tworzący cykl). Stąd tożsamość równania (4.6) jest równoznaczna z wystąpieniem cyklu.

c.k.d.

W kontekście systemów transportowych przedstawione lematy należy interpretować następująco: Jeśli przyjęte reguły priorytetowania Θ i stan początkowy S_0 doprowadzą do sytuacji, w której w systemie pojawi się stan S_i reprezentowany przez graf żądań zasobowych G^{si}, dla którego równanie stanu (4.6) jest tożsamościowe, to stan S_i zawiera cykl (co oznacza, że system jest w stanie blokady częściowej lub całkowitej). Konsekwencją Lematów 4.1, 4.2 i 4.3 jest Lemat 4.4:

Lemat 4.4

Dany jest stan osiągalny S_i reprezentowany przez graf żądań zasobowych G^{si}, dane są marszruty procesów P_i, reguły priorytetowania Θ, grafy $G^{s(i+1)}$, $G^{s(i+2)}$, ..., $G^{s(i-k-1)} \in Gp^{**}$, które spełniają ograniczenia (4.4), (4.5). Stan S_{i+k} reprezentowany jest przez graf żądań zasobowych $G^{s(i-k)} \in Gp^{**}$, gdzie współrzędne końcowe łuków opisuje równanie (4.11).

Jeżeli istnieje taki graf żądań zasobowych G^{sj}, gdzie $i \le j \le i+k$, reprezentujący osiągalny stan pośredni S_j między stanami S_i i S_{i+k}, który zawiera podgraf ${}^{z}G^{si}$ będący cyklem, to ogólne równanie stanu (4.11) jest tożsamościowe lub sprzeczne.

Dowód

Jeżeli G^{sj} zawiera podgraf ${}^{z}G^{si}$ będący cyklem to równanie stanu (4.8) opisujące ten graf zgodnie z Lematem 4.2 jest tożsamościowe. Wówczas ogólne równanie (4.11), którego argumentem jest równanie stanu grafu G^{sj}, jest też tożsamościowe.

c.k.d.

Lemat 4.4 prowadzi do następujących wniosków: Jeżeli system transportowy jest opisany sekwencją stanów $S = (S_0, S_1, \ldots, S_i, \ldots, S_w)$ gdzie znany jest stan początkowy S_0, a S_i jest stanem osiągalnym i jest to stan blokady (graf G^{si} zawiera podgraf będący cyklem), to

ogólne równanie stanu (4.12) (będące szczególnym przypadkiem (4.11)), umożliwiające wyznaczenie stanu S_w (współczynników grafu G^W), jest sprzeczne lub tożsamościowe. Lemat 4.4 dla równań stanu o charakterze tożsamościowym jest również spełniony w kierunku odwrotnym:

Lemat 4.5

Dany jest stan osiągalny S_i reprezentowany przez graf żądań zasobowych G^{si}. Dane są marszruty procesów P_i, reguły priorytetowania Θ, grafy $G^{s(i+1)}$, $G^{s(i+2)}$, ... , $G^{s(i+k-1)} \in Gp^{**}$, które spełniają ograniczenia (4.4), (4.5). Stan przyszły S_{i+k} reprezentowany jest przez graf żądań zasobowych $G^{s(i+k)} \in Gp^{**}$, gdzie współrzędne końcowe łuków opisuje równanie (4.11).

Jeżeli równanie stanu (4.11) dla stanów S_i, S_{i+k}, jest tożsamościowe, to istnieje taki graf żądań zasobowych G^{sj} (w łańcuchu grafów żądań zasobowych G), gdzie $i \le j \le i+k$, reprezentujący stan osiągalny S_j, pośredni między stanami S_i i S_{i-k}, który zawiera podgraf ${}^zG^{si}$ będący cyklem.

Dowód

Równanie (4.11) jest równaniem tożsamościowym w przypadku gdy istnieje co najmniej jedno równanie tożsamościowe ${}^p x_v^{s(i+1)}(x)$, wchodzące w skład tego równania. Zgodnie z Lematem 4.3, równanie ${}^p x_v^{s(i+1)}(x)$ (wyrażone w postaci (4.8)) jest tożsamościowe gdy opisuje procesy reprezentowane przez graf ${}^zG^{si}$ będący cyklem. Zatem jeżeli równanie (4.11) jest tożsamościowe to jeden z grafów opisywany przez argumenty równania (4.11) jest cyklem.

c.k.d.

Lemat 4.5 prowadzi do następującego Twierdzenia 4.1:

Twierdzenie 4.1

Dany jest system transportowy, opisany marszrutami P. Znany jest stan początkowy S_0, znane są reguły priorytetowania Θ. Stany systemu S są reprezentowane przez sekwencje grafów żądań zasobowych.

Jeżeli ogólne równanie stanu (4.12) jest tożsamościowe to dany system transportowy, w oknie czasowym W, osiągnie stan blokady częściowej bądź całkowitej.

Dowód

Równanie (4.12) jest szczególnym przypadkiem równania (4.11). Zgodnie z Lematem 5, jeśli równanie (4.12) jest tożsamościowe, to istnieje graf G^{sj} zwierający podgraf będący cyklem. Grafy G^{si} stanowią reprezentację stanów S_i osiąganych przez system w oknie czasowym W. Zatem jeżeli równanie (4.12) jest tożsamościowe, to w oknie czasowym wystąpi stan blokady.

c.k.d.

W oparciu o Twierdzenie 4.1 możliwe jest poszukiwanie warunków systemu, które gwarantują bezkolizyjną i bezblokadową pracę wózków samojezdnych. Twierdzenie to jest konsekwencją przyjętych ograniczeń (4.2), (4.3), (4.4), (4.5), stanowiących reprezentację założeń charakteryzujących rozważaną klasę systemów transportowych. Ograniczenia opisują

relacje zachodzące między stanem początkowym S_0, regułami priorytetowania Θ i sekwencją terminów rozpoczęcia operacji x.

Interpretacja twierdzenia jest następująca: jeżeli dla zadanego stanu początkowego S_0 i reguł priorytetowania Θ, w wyniku spełnienia ograniczeń (4.2), (4.3), (4.4), (4.5), istnieją elementy sekwencji x, które nie posiadają jednoznacznej wartości, to w systemie osiągnięty zostanie stan blokady.

Na uwagę zasługuje więc fakt, że jednej parze S_0 i Θ odpowiada tylko jedna sekwencja x gwarantująca spełnienie założeń stawianych klasie sytemu. Dla zadanej postaci sekwencji S_0 i Θ, odpowiada sekwencja x, której elementy są wyznaczane z ogólnych równań stanu (4.12). Równania te oparte są o równania stanu (4.8), dla których jeśli istnieje rozwiązanie, to jest to rozwiązanie pojedyncze (terminowi rozpoczęcia operacji x przyporządkowana jest jedna wartość). W związku z tym jeśli równanie (4.12) nie jest tożsamościowe to posiada tylko jedno rozwiązanie (jedna postać sekwencji x). Innymi słowy dla zadanego stanu początkowego i reguł priorytetowania istnieje tylko jeden harmonogram pracy wózków gwarantujący bezkolizyjną i bezblokadową pracę systemu.

W praktycznym przypadku, wyznaczanie wartości x (a tym samym określenie czy system osiągnie stan blokady) dla zadanych postaci sekwencji S_0 i Θ sprowadza się do rozwiązania q-elementowego układu równań postaci (4.12), gdzie q określa liczbę procesów realizowanych w systemie. Jeżeli w wyniku rozwiązania układu otrzymane zostanie jedno rozwiązanie sekwencji x, to system nie osiągnie w oknie W stanu blokady, w przypadku wielu rozwiązań sekwencji x system osiągnie w oknie czasowym W stan blokady częściowej lub całkowitej. Każde równanie układu stanowi wyrażenie w skład, którego wchodzi w elementów ${}^{p}x_v^{sw}(x)$, gdzie w określa liczę stanów realizowanych w oknie czasowym W. Elementy ${}^{p}x_v^{si}(x)$ wyznaczane są między innymi z równań (zgodnie z (4.8)) opisanych przez qz operacji max (qz - liczba łuków w podgrafie ${}^{z}G^{si}$).

Do rozwiązania układu, a tym samym do wyznaczenia wartości elementów sekwencji x, należy wykonać $q \cdot w \cdot qz$ operacji max. Przyjmując, że z jedną operacją max związany jest jeden krok obliczeniowy oraz, że $w = n$ (n – liczba operacji elementarnych realizowanych w systemie, w trakcie trwania okna W) i $qz = q$, liczba kroków obliczeniowych Z_x konieczna do wyznaczenia wartości elementów sekwencji x wynosi:

$$Z_x = q^2 n, \tag{4.14}$$

gdzie: q – liczba procesów realizowanych w systemie,

n – liczba operacji elementarnych realizowanych w systemie w trakcie trwania okna W.

Zależność (4.14) stanowi górne oszacowanie liczby operacji koniecznych do stwierdzenia czy dla zadanych parametrów S_0 i Θ system osiągnie stan blokady. Należy zaznaczyć, że jest to problem o złożoności wielomianowej.

W rozważanym przypadku (dla pytania postawionego w punkcie 4.1) poszukiwane są jednak takie postacie warunków S_0 i Θ, dla których system będzie się charakteryzował

brakiem blokady i kolizji. Zwykle odbywa się to poprzez sprawdzanie dla wszystkich wartości S_0 i Θ, czy system nie osiąga stanu blokady (poprzez rozwiązanie q-elementowego układu równań opisanego w postaci równań (4.10)). Dla wyznaczonego w ten sposób zbioru sekwencji S_0 i Θ, określane są w dalszej kolejności związki (w postaci relacji) charakterystyczne dla tych zbiorów. Związki te wyrażone w postaci zdań logicznych stanowią warunki gwarantujące brak blokad lub kolizji w systemie.

Przeszukiwanie przestrzeni wartości S_0, Θ, pod kątem istnienia harmonogramów bezblokadowych odpowiada niejako przeszukiwaniu metodą prób i błędów. Otrzymane wartości zmiennych są analizowane pod kątem istnienia relacji między zmiennymi S_0, Θ, które zagwarantują spełnienie żądanej właściwości.

Wraz ze wzrostem rozmiaru systemu transportowego (większa liczba procesów, zasobów, większy rozmiar marszrut) rośnie rozmiar sekwencji S_0 i Θ. W przypadku sekwencji stanów początkowych S_0, przestrzeń potencjalnych wartości rośnie w sposób wykładniczy wraz ze wzrostem rozmiaru sekwencji (ks^q, ks – liczba zasobów, q – rozmiar sekwencji S_0). Podobnie jest w przypadku sekwencji Θ, każdej regule priorytetowania σ_j odpowiada przestrzeń potencjalnych wartości, której rozmiar rośnie permutacyjnie (funkcja typu silnia) wraz ze wzrostem rozmiaru sekwencji σ_j ($ws_j!$ – liczba możliwych kolejności obsługi procesów na zasobie R_j, ws_j – rozmiar sekwencji σ_j).

Ze względu na to, że zależność rozmiaru przestrzeni potencjalnych wartości od rozmiaru sekwencji S_0 i Θ nie ma wielomianowego charakteru, problem wyznaczenia właściwości systemu, których spełnienie gwarantuje brak blokady, jest zatem problemem NP-trudnym. W praktycznych problemach rozmiar przestrzeni potencjalnych wartości jest tak duży, że niemożliwe jest jej przeszukiwanie stosując przegląd zupełny. W tym celu wykorzystuje się metody umożliwiające automatyczne poszukiwanie takich właściwości systemu oparte na technikach programowania z ograniczeniami implementujących metodę logiczno-algebraiczną.

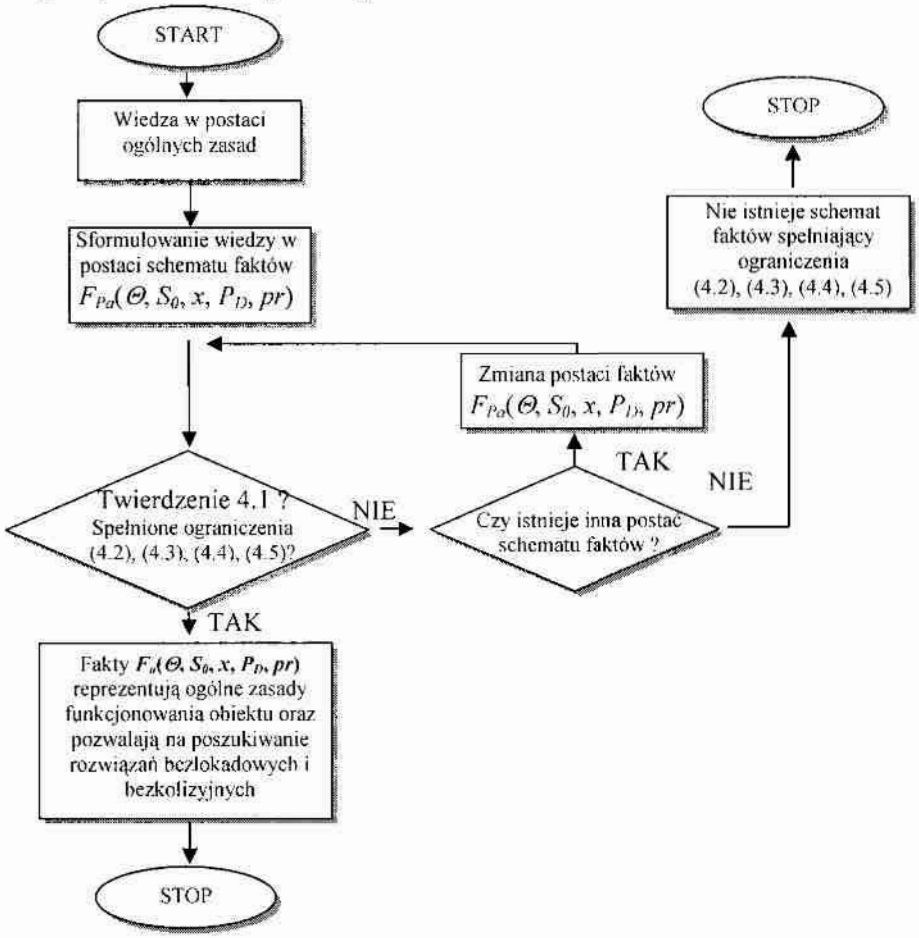
Twierdzenie 4.1 stanowi opis związków jakie muszą zaistnieć między zmiennymi x, S_0 i Θ modelu SWPC, aby w systemie wystąpił stan blokady. Twierdzenie to wraz z ograniczeniami (4.2), (4.3), (4.4), (4.5), stanowi jednocześnie podstawę do budowy schematu faktów opisujących obiekty w postaci systemów transportowych rozważanej klasy. W kontekście zagadnień obejmujących proces weryfikacji bazy wiedzy przedstawionych w punkcie 4.1, wyznaczone twierdzenie odpowiada realizacji drugiego zagadnienia (wyznaczenie relacji, między zmiennymi modelu SWPC, opisującej spełnienie właściwości wyjściowej – stan kolizji i blokady). Kolejny rozdział dotyczy budowy schematu faktów w oparciu o sformułowane związki – Twierdzenie 4.1 i ograniczenia (4.2), (4.3), (4.4), (4.5).

4.2.4. Warunki wystarczające

Wyprowadzone w poprzednim punkcie Twierdzenie 4.1 jest słuszne przy założeniu, że spełnione są ograniczenia (4.2), (4.3), (4.4), (4.5), (grafy G^{si} spełniają żądania zasobowe). Ograniczenia te stanowią więc swoiste warunki wystarczające, których spełnienie umożliwia

wyznaczenie rozwiązań bezblokadowych. Opisują one relacje zachodzące pomiędzy elementami sekwencji x, a więc są one warunkami wystarczającymi w kontekście tych zmiennych. W tego typu systemach poszukuje się jednak warunków gwarantujących bezblokadową pracę, będących opisem relacji w kontekście zmiennych S_0, Θ (poszukiwanie właściwości wejściowej $Fu(S_0,\Theta)$ układu z rysunku 4.3). Postać S_0 i Θ wyznacza się rozwiązując problem decyzyjny. Jednak by móc prowadzić wnioskowanie umożliwiające wyznaczenie tych własności, konieczne jest posiadanie reprezentacji wiedzy (KB), a ściślej mówiąc schematu faktów $F_{Pa}(\Theta, S_0, x, P_D, pr)$, gdzie: pr – zmienne parametryczne. Wymagane jest by schemat faktów zawierał wiedzę gwarantującą otrzymanie rozwiązań bezblokadowych na etapie wnioskowania.

Inaczej mówiąc, zbiór faktów musi umożliwić korzystanie z Twierdzenia 4.1. Twierdzenie to zatem stanowi swego rodzaju ograniczenie określające postać faktów opisujących wiedzę o systemie transportowym.



Rys. 4.20. Procedura budowy faktów $F_a(\Theta, S_0, x, P_D)$

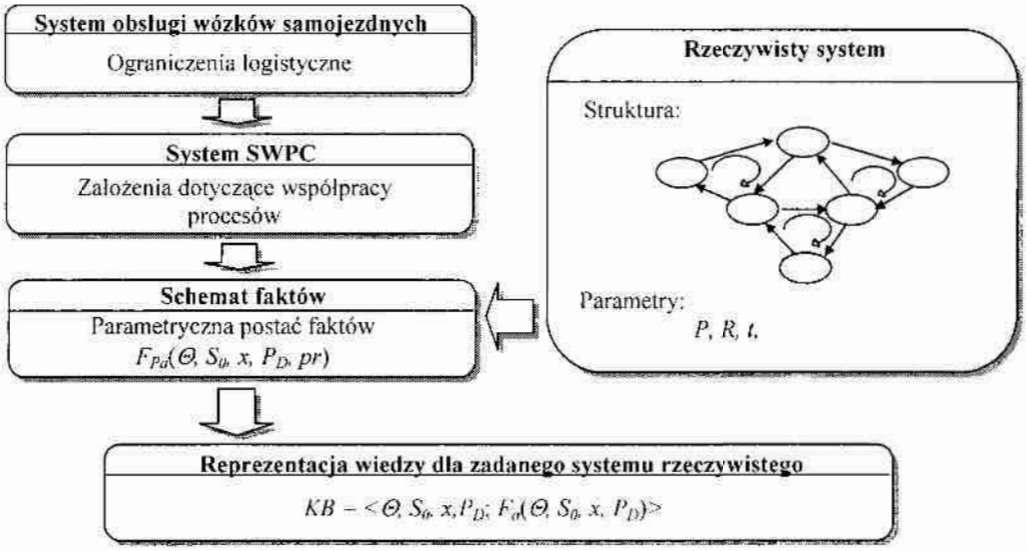
Wiedza ogólna (założenia dla klasy systemów) wyrażana jest w postaci faktów $F_a(\Theta, S_0, x, P_D)$. Na rysunku 4.20 przedstawiono procedurę budowy schematu faktów $F_{Pa}(\Theta, S_0, x, P_D, pr)$. Jeżeli fakty sformułowane dla zadanego zbioru wartości parametrów pr, nie gwarantują spełnienia ograniczeń (4.2), (4.3), (4.4), (4.5), (a tym samym gwarantują słuszność Twierdzenia 4.1), to należy zmienić postać faktów i ponownie sprawdzić czy nowe fakty

odpowiadają ograniczeniom (4.2), (4.3), (4.4), (4.5). Procedurę należy powtarzać aż do momentu uzyskania postaci faktów implikującej spełnienie ograniczeń (4.2), (4.3), (4.4), (4.5).

Efektem przedstawionej procedury jest tzw. ogólna postać faktów uniwersalnych (schematu faktów), opisujących dowolny system współbieżnych procesów cyklicznych. Zbiór schematu faktów ma postać (zgodnie z podejściem przedstawionym w rozdziale 3): $F_{Pa}(\Theta, S_0, x, P_D, pr) = \{F_{1a}(\Theta, S_0, x, P_D, pr), F_{2a}(\Theta, S_0, x, P_D, pr), \ldots, F_{Ka}(\Theta, S_0, x, P_D, pr)\}$. Korzystając ze schematu faktów możliwe jest automatyczne generowanie reprezentacji wiedzy.

Idea automatycznego generowania reprezentacji wiedzy dla arbitralnie zadanego systemu obsługi wózków samojezdnych, przedstawiona została na rysunku 4.21. Systemy obsługi wózków samojezdnych opisane ograniczeniami logistycznymi, można przedstawiać w postaci systemów współbieżnych procesów cyklicznych. W oparciu o ograniczenia (4.2), (4.3), (4.4), (4.5), przedstawione w sekcji 4.3, możliwe jest sformułowanie ogólnej postaci faktów $F_{Pa}(\Theta, S_0, x, P_D, pr)$, postaci zależnej od wartości parametrów pr systemu. Taka ogólna postać faktów stanowi **schemat faktów** systemu transportowego.

Schemat faktów stanowi swoisty szkielet, który uzupełniany o parametry pr konkretnego systemu, pozwala na budowę reprezentacji wiedzy KB opisującej właściwości (zachowanie) systemu obsługi wózków samojezdnych. Dla każdego systemu współbieżnych procesów cyklicznych, w skład którego wchodzi q procesów, k zasobów znane są sekwencje P, t oraz R – stanowią one podstawowe parametry systemu transportowego. Ponadto definiowane są sekwencje dodatkowe zmiennych decyzyjnych (parametry pomocnicze) i funkcje określające zachowanie się systemu, umożliwiające budowę schematu faktów.



Rys. 4.21. Proces generowania reprezentacji wiedzy dla określonego systemu w oparciu o schemat faktów

Wyszczególniane są parametry: ko – liczba zasobów lokalnych, ks – liczba zasobów współdzielonych, w_i – liczba procesów obsługiwanych przez i-ty zasób współdzielony.

Procesy obsługiwane przez *i-ty* zasób współdzielony są przedstawiane w postaci sekwencji Ws_i:

$$Ws_i = (P_j, P_l, \ldots, P_o),\ i = 1,2, \ldots, ks;\ P_j, P_l, \ldots, P_o \in P;\ \|Ws_i\| = w_i, \qquad (4.15)$$

Sekwencja terminów rozpoczęcia elementarnych operacji, procesów obsługiwanych przez *i*-ty zasób definiowana jest w postaci:

$$No_i = (x_j, x_k, \ldots, x_c),\quad i = 1,2, \ldots, ks;\ x_j, x_k, \ldots, x_c \in x;\ \|No_i\| = w_i, \qquad (4.16)$$

Sekwencja No_i określa terminy rozpoczęcia poszczególnych operacji przez odpowiadający im proces na *i*-tym zasobie.

Sekwencja Sp_i jest sekwencją, której elementy określają procesy obsługiwane na *i*-tym zasobie przed innymi procesami.

$$Sp_i = (P_j, P_l, \ldots, P_o),\ i = 1,2, \ldots, kp, \qquad (4.17)$$

gdzie: crd_jSp_i – oznacza proces, który na zasobie R_i jest obsługiwany przed procesem crd_jWs_i,

$crd_jSp_i = \Lambda$ – oznacza, że proces crd_jWs_i jest obsługiwany na zasobie R_j jako pierwszy.

Sekwencje Sp_i i Ws_i wykorzystywane są do reprezentowania reguł prioryterowania σ_i. Dla zasobu współdzielonego R_i definiuje się postać sekwencji Ws_i (która stanowi listę obsługiwanych na zasobie R_i procesów). Reguła σ_i jest jedną z permutacji elementów sekwencji Ws_i. W zależności od postaci reguły σ_i oraz przyjętej postaci sekwencji Ws_i elementy sekwencji Sp_i są opisane następująco:

$$crd_jSp_i = \begin{cases} crd_{ind\{crd_jWs_i, \sigma_i\}-1}\sigma_i & \text{dla } ind\{crd_jWs_i, \sigma_i\} > 1 \\ \Lambda & \text{dla } ind\{crd_jWs_i, \sigma_i\} = 1 \end{cases}$$

gdzie: $ind\{crd_jWs_i,\ \sigma_i\}$ – operator indeksu wynikiem, którego jest indeks elementu crd_jWs_i w sekwencji σ_i. Operator *ind* definiowany jest następująco:

$ind\{a, E\} = b \Leftrightarrow crd_bE = a$,

a – element sekwencji E,

b – indeks a-tego elementu w sekwencji E

Na przykład, zakładając, że $Ws_1 = (P_1, P_2, P_3, P_4)$ to sekwencji $\sigma_1 = (P_2, P_3, P_1, P_4)$ odpowiada sekwencja $Sp_1 = (P_3, \Lambda, P_2, P_1)$.

Sekwencja Sp_i wykorzystywana jest do budowy faktów odpowiadających ograniczeniu (4.5). Dodatkowo definiuje się operatory:

$$\sim A\{i\} = \begin{cases} \text{"="} \text{ jeżeli } x_i \text{ jest terminem rozpoczęcia operacji na zasobie lokalnym} \\ \text{"}\geq\text{"} \text{ jeżeli } x_i \text{ jest terminem rozpoczęcia operacji na zasobie współdzielonym} \end{cases}$$

$$WR\{R_a, b\} = \begin{cases} x_{wr} & \text{jeżeli operacja procesu } crd_bWs_a \text{ nie jest ostatnią operacją na zasobie } R_a \\ 0 & \text{jeżeli operacja procesu } crd_bWs_a \text{ jest ostatnią operacją na zasobie } R_a \end{cases}$$

gdzie: x_{wr} – termin rozpoczęcia operacji występującej na zasobie R_a po ukończeniu realizacji operacji procesu crd_bWs_a na zasobie R_a.

$$PR\{R_a, b\} = \begin{cases} x_{pr} & \text{jeżeli operacja procesu } crd_bWs_a \text{ nie jest pierwsza z operacji} \\ & \text{wykonywanych na zasobie } R_a \\ 0 & \text{jeżeli operacja procesu } crd_bWs_a \text{ jest pierwszą z operacji} \\ & \text{wykonywanych na zasobie } R_a \end{cases}$$

gdzie: $x_{pr} = x_{pr}' + t_{pr}'$ – termin ukończenia operacji występującej na zasobie R_a przed rozpoczęciem operacji procesu crd_bWs_a na zasobie R_a.

$$D\{i\} = \begin{cases} \sum_{k=1}^{i-1} m_k & \text{jeżeli } i > 1 \\ 0 & \text{jeżeli } i = 1 \end{cases}$$

gdzie: m_k – liczba operacji elementarnych realizowanych przez k-ty proces: $m_k = \|P_q\|$.

Wprowadzone operatory stanowią wyrażenia pomocnicze, wykorzystywane do budowy faktów chodzących w skład formułowanego schematu faktów. Schemat faktów ma postać:

$$F_{Pa}(\Theta, S_0, x, P_D, pr) = \{F_{1a}(\Theta, S_0, x, P_D, pr), \ldots, F_{5a}(\Theta, S_0, x, P_D, pr)\}. \qquad (4.18)$$

- Ograniczenia kolejnościowe (4.2), (4.3) opisane są przez fakty postaci:
  $F_{1a}(\Theta, S_0, x, P_D, pr)$: $(crd_kS_0 = crd_jP_i) \Rightarrow (x_{[D\{i\}+j]} = 0) \wedge (x_{[D\{i\}+j+1]} \sim A\{D\{i\}+j+1\}$ $x_{[D\{i\}+j]} + t_{[D\{i\}+j]}) \wedge \ldots \wedge (x_{[D\{i\}+mi]} \sim A\{D\{i\}+m_i\}$ $x_{[D\{i\}+mi-1]} + t_{[D\{i\}+mi-1]}) \wedge$ $(x_{[D\{i\}+1]} \sim A\{D\{i\}+1\}$ $x_{[D\{i\}+mi]} + t_{[D\{i\}+mi]}) \wedge \ldots \wedge (x_{[D\{i\}+j-1]} \sim A\{D\{i\}+j-1\}$ $x_{[D\{i\}+j-2]} + t_{[D\{i\}+j-2]})$
  $j = 1,2, \ldots, m_i$; $i = 1,2, \ldots, q$
  Przedstawiony fakt należy interpretować następująco: Jeśli określony proces P_i rozpoczyna swoją pracę od pewnego zasobu R_y (określonego w sekwencji S_0: $R_y = crd_kS_0$) to termin rozpoczęcia operacji procesu P_i na zasobie R_y jest równy: $x_{[D(i)+j]} = 0$. Wartości terminów kolejnych operacji są równe lub większe od wartości ukończenia operacji je poprzedzających. Tego typu fakty budowane są dla wszystkich realizowanych procesów.
- Zasady dotyczące zachowania się procesów na zasobach (ograniczenia (4.4), (4.5)) opisane są przez fakty:
  Dla $ki = 2, \ldots, w_i$; $li = 1,2, \ldots, w_i$; $i = 1,2, \ldots, ks$;
  $F_{2a}(\Theta, S_0, x, P_D, pr)$: $(crd_1\sigma_i = crd_{li}Ws_i) \Rightarrow (crd_{li}Sp_i = \Lambda)$,
  $F_{3a}(\Theta, S_0, x, P_D, pr)$: $(crd_{ki}\sigma_i = crd_{li}Ws_i) \Rightarrow (crd_{li}Sp_i = crd_{ki-1}\sigma_i)$,
  Dla $ki = 1, \ldots, wi$; $li = 1,2, \ldots, wi$; $i = 1,2, \ldots, ks$;
  $F_{4a}(\Theta, S_0, x, P_D, pr)$: $(crd_{ki}Sp_i = \Lambda) \Rightarrow (crd_{ki}No_i = PR\{R_i, k\})$,
  $F_{5a}(\Theta, S_0, x, P_D, pr)$: $(crd_{ki}Sp_i = crd_{li}Ws_i) \Rightarrow (crd_{ki}No_i = \max\{WR\{R_i, li\}, PR\{R_i, k\}\})$,

gdzie: $P_D = \{No_1, \ldots, No_{ks}, Sp_1, \ldots, Sp_{ks}, Ws_1, \ldots, Ws_{ks}\}$ – zbiór zmiennych pomocniczych, których wartości są wyznaczane ze zmiennych Θ, S_0, x, pr, (zmienne pomocnicze nie są zadawane przez użytkownika),

$pr = \{P_1, \ldots, P_q, t_1, \ldots, t_n, R_1, R_2, \ldots, R_k\}$ – zbiór parametrów.

Przedstawione fakty odpowiadają intuicji ograniczeń (4.4), (4.5). Dla określonego zasobu współdzielonego R_i opisanego regułą priorytetowania σ_i określane są wartości terminów rozpoczęcia operacji No na tym zasobie. Termin rozpoczęcia określonej operacji $crd_{ki}No_i$ przez pewien proces P_l, wyznaczany jest jako wartość maksymalna z wartości operatorów $WR\{R_i,li\}$, $PR\{R_i,k\}$: $\max\{WR\{R_i,li\}, PR\{R_i,k\}\}$.

Tak sformułowana postać schematu faktów umożliwia automatyczne generowanie „szczegółowej" reprezentacji wiedzy KB dla dowolnej struktury systemu współbieżnych procesów cyklicznych.

Przykład 4.7. Reprezentacja wiedzy dla przykładowego systemu SWPC

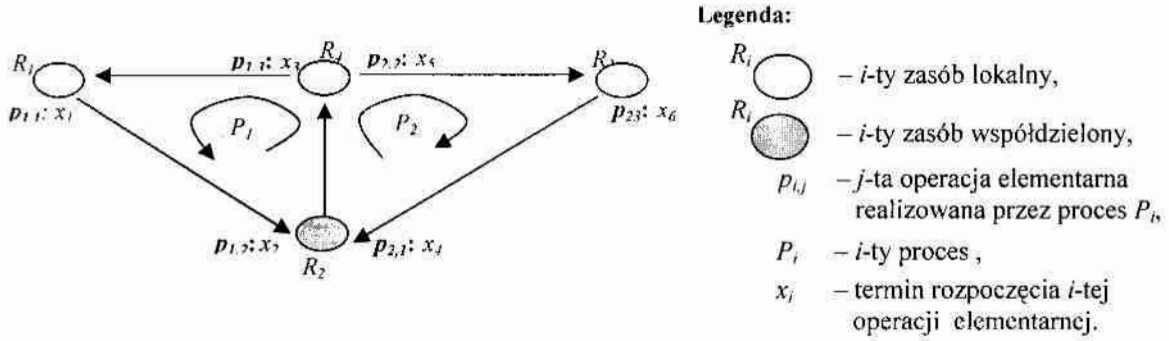
Celem przykładu jest ilustracja reprezentacji wiedzy dla zadanego systemu SWPC.

Dany jest system współbieżnych procesów cyklicznych (rysunek 4.22), modelujący strukturę pewnego systemu obsługi wózków samojezdnych. Należy wyznaczyć reprezentację KB tego systemu.

System opisany jest następującymi parametrami:

$P_1 = (R_1, R_2, R_4)$, $P_2 = (R_2, R_4, R_3)$, $R = (R_1, R_2, R_3, R_4)$,

$x = (x_1, x_2, x_3, x_4, x_5, x_6)$, $t = (2, 2, 2, 1, 3, 2)$.



Rys. 4.22. System współbieżnych procesów cyklicznych

Sekwencje Ws, No mają postać:

$$Ws_1 = (P_1, P_2),\ Ws_2 = (P_1, P_2),\ No_1 = (x_2, x_4),\ No_2 = (x_5, x_6).$$

Sekwencje σ_1, σ_2, Sp_1, Sp_2, S_0 są sekwencjami 2 elementowymi.

$$\sigma_1 = (P_j, P_k),\ \sigma_2 = (P_l, P_r),\ Sp_1 = (P_o, P_r),\ Sp_2 = (P_q, P_e),\ S_0 = (R_i, R_j)$$

gdzie: $\Theta = (\sigma_1, \sigma_2)$, $\Theta \in O$, $Sp = (Sp_1, Sp_2)$, $Sp \in SP$

Dla przedstawionej struktury reprezentacja wiedzy przyjmuje postać:

$KB = < S0,\ O, X,\ SP;\ Re >$

gdzie: $Re = \{(S_0,\ \Theta,\ x,\ Sp)\colon Q(S_0,\ \Theta,\ x,\ Sp) = \bar{1}\}$.

W oparciu o sformułowany schemat, zbudowany zestaw faktów $F(S_0,\ \Theta,\ x,\ Sp)$ ma postać:

Fakty dotyczące czasów rozpoczęcia operacji procesów P_1, P_2:

$$F_1\colon (crd_1S_0 = R_1) \Rightarrow (x_1 = 0) \wedge (x_2 \geq x_1 + t_1) \wedge (x_3 \geq x_2 + t_3),$$
$$F_2\colon (crd_1S_0 = R_2) \Rightarrow (x_2 = 0) \wedge (x_3 \geq x_2 + t_2) \wedge (x_1 = x_3 + t_3),$$
$$F_3\colon (crd_1S_0 = R_4) \Rightarrow (x_3 = 0) \wedge (x_1 = x_3 + t_3) \wedge (x_2 \geq x_1 + t_1),$$
$$F_4\colon (crd_2S_0 = R_2) \Rightarrow (x_4 = 0) \wedge (x_5 \geq x_4 + t_4) \wedge (x_3 = x_5 + t_5),$$
$$F_5\colon (crd_2S_0 = R_4) \Rightarrow (x_5 = 0) \wedge (x_6 = x_5 + t_5) \wedge (x_4 \geq x_6 + t_6),$$
$$F_6\colon (crd_2S_0 = R_3) \Rightarrow (x_6 = 0) \wedge (x_4 \geq x_6 + t_6) \wedge (x_5 \geq x_4 + t_4).$$

Fakty dotyczące czasów rozpoczęcia operacji na zasobach R_2, R_4:

$$F_7\colon (crd_1\sigma_1 = P_1) \Rightarrow (crd_1Sp_1 = \Lambda),$$
$$F_8\colon (crd_1\sigma_1 = P_2) \Rightarrow (crd_2Sp_1 = \Lambda),$$
$$F_9\colon (crd_2\sigma_1 = P_1) \Rightarrow (crd_1Sp_1 = crd_1\sigma_1),$$
$$F_{10}\colon (crd_2\sigma_1 = P_2) \Rightarrow (crd_2Sp_1 = crd_1\sigma_1),$$
$$F_{11}\colon (crd_1Sp_1 = \Lambda) \Rightarrow (x_2 = PR(R_1,1)),$$
$$F_{12}\colon (crd_2Sp_1 = \Lambda) \Rightarrow (x_4 = PR(R_1,2)),$$
$$F_{12}\colon (crd_1Sp_1 = P_1) \Rightarrow (x_2 = \max\{WR(R_1,1), PR(R_1,1)\}),$$
$$F_{13}\colon (crd_2Sp_1 = P_1) \Rightarrow (x_4 = \max\{WR(R_1,1), PR(R_1,2)\}),$$
$$F_{14}\colon (crd_1Sp_1 = P_2) \Rightarrow (x_2 = \max\{WR(R_1,2), PR(R_1,1)\}),$$
$$F_{15}\colon (crd_2Sp_1 = P_2) \Rightarrow (x_4 = \max\{WR(R_1,2), PR(R_1,2)\}),$$
$$F_{16}\colon (crd_1\sigma_2 = P_1) \Rightarrow (crd_1Sp_2 = \Lambda),$$
$$F_{17}\colon (crd_1\sigma_2 = P_2) \Rightarrow (crd_2Sp_2 = \Lambda),$$
$$F_{18}\colon (crd_2\sigma_2 = P_1) \Rightarrow (crd_1Sp_2 = crd_1\sigma_2),$$
$$F_{19}\colon (crd_2\sigma_2 = P_2) \Rightarrow (crd_2Sp_2 = crd_1\sigma_2),$$
$$F_{20}\colon (crd_1Sp_2 = \Lambda) \Rightarrow (x_5 = PR(R_2,1)),$$
$$F_{21}\colon (crd_2Sp_2 = \Lambda) \Rightarrow (x_6 = PR(R_2,2)),$$
$$F_{22}\colon (crd_1Sp_2 = P_1) \Rightarrow (x_5 = \max\{WR(R_2,1), PR(R_2,1)\}),$$
$$F_{23}\colon (crd_2Sp_2 = P_1) \Rightarrow (x_6 = \max\{WR(R_2,1), PR(R_2,2)\}),$$
$$F_{24}\colon (crd_1Sp_2 = P_2) \Rightarrow (x_5 = \max\{WR(R_2,2), PR(2,1)\}),$$
$$F_{25}\colon (crd_2Sp_2 = P_2) \Rightarrow (x_6 = \max\{WR(R_2,2), PR(R_2,2)\}),$$

Wyznaczone fakty $F_1 - F_{25}$ stanowią zestaw w oparciu, o który możliwe jest poszukiwanie warunków wystarczających gwarantujących realizację wszystkich procesów w cyklach nie przekraczających założonego terminu H.

■

Przedstawiony schemat faktów $F_{Pa}(\Theta, S_0, x, P_D, pr)$ stanowi jedną z wielu możliwych reprezentacji relacji (ograniczenia (4.2), (4.3), (4.4), (4.5) oraz Twierdzenie 4.1) opisujących obiekt (system transportowy). Z tej perspektywy prezentowany schemat faktów odpowiada realizacji zagadnienia 3 weryfikacji bazy wiedzy przedstawionego w punkcie 4.1. Kolejne zagadnienie (zagadnienie 4 z punktu 4.1) obejmuje już wykorzystanie sformułowanego schematu faktów do wyznaczania warunków wystarczających.

Dysponując zdefiniowanym schematem faktów możliwe jest, dla różnych wartości parametrów systemu transportowego (R, t, P_i), automatyczne wyznaczanie postaci reprezentacji wiedzy KB.

W oparciu o wyznaczoną reprezentację prowadzone jest wnioskowanie mające na celu wyznaczenie właściwości wejściowej $Fu(S_0, \Theta)$, która gwarantuje spełnienie żądanej właściwości wyjściowej $Fy(x)$. W tym celu rozwiązywany jest problem decyzyjny.

W rozdziale 3 przedstawiona została idea formułowania problemu decyzyjnego w postaci dwóch problemów klasy PSO (PSO_{Su1}, PSO_{Su2}). Możliwe jest zatem wykorzystanie technik programowania z ograniczeniami (a tym samym ich zalet) do rozwiązywania tego typu problemu. Jednak podejście oparte tylko na wykorzystaniu mechanizmów propagacji ograniczeń i dystrybucji zmiennych okazuje się w wielu przypadkach niewystarczające do rozwiązania problemu decyzyjnego w akceptowalnym (przez użytkownika) czasie. Z tego też względu powstaje potrzeba poszukiwania czasowo efektywnych strategii przeszukiwania potencjalnej przestrzeni rozwiązań. W kolejnym rozdziale omówiono podejście poszukiwania rozwiązań oparte o częściowe przeszukiwanie przestrzeni potencjalnych rozwiązań.

4.3. Strategie przeszukiwania

Przedstawiona na rysunku 2.4 struktura interakcyjnego systemu wspomagania decyzji zawiera **Moduł efektywnych strategii przeszukiwania**. Jest on odpowiedzialny za dobór strategii poszukiwania rozwiązania PSO, oraz dobór strategii wyznaczania warunków wystarczających (w tym przypadku strategii rozwiązywania problemu decyzyjnego i weryfikacji spójności bazy wiedzy). O ile literatura [81], [53], [29], [9], bogata jest w opisy różnego rodzaju strategii poszukiwania rozwiązań dopuszczalnych, o tyle tematyka dotycząca wyznaczania warunków wystarczających (a tym samym strategii poszukiwania takich warunków) nie cieszy się popularnością.

Przez strategie poszukiwania rozwiązania należy rozumieć sposób, w jaki następuje badanie drzewa potencjalnych rozwiązań. Dąży się do tego, by wykorzystywane strategie maksymalnie ograniczały czas poszukiwania rozwiązania. W przedstawionym kontekście, wyróżnia się dwie podstawowe grupy strategii. Pierwszą z nich są strategie określające sposób „poruszania" się w przestrzeni potencjalnych rozwiązań. Do najczęściej stosowanych można zaliczyć te, które bazują przede wszystkim na poszukiwaniu zwanym *„okrajaniem drzewa" SBS* (ang. Slice Based Search), a także te wykorzystujące zasadę *„najpierw w głąb" DFS* (ang. Depth First Search).

Strategia *Depth First Search* polega na analizie pierwszej gałęzi drzewa, począwszy od jego wierzchołka, aż do ostatniej gałęzi (liścia). Procedura poszukiwania powtarzana jest dla kolejnych gałęzi drzewa. Jej realizacja sprowadza się do podstawiania dziedzin zmiennych decyzyjnych począwszy od wartości najmniejszych.

Interleaved Depth First Search (*IDFS*) jest procedurą poszukiwania bazującą na strategii *DFS*. Przy użyciu strategii *IDFS* analizowane są wszystkie punkty wyboru (potencjalne rozwiązania) danego poziomu drzewa, a następnie proces poszukiwania odbywa się na kolejnym poziomie. Podejście takie pozwala uniknąć zbędnych ścieżek, tzn. gałęzi, które nie zawierają rozwiązania.

Istotą strategii przeszukiwania okrojonego (*SBS*) jest założenie, że jeżeli określona heurystyka nie znajduje rozwiązania, oznacza to, że rozwiązanie zostałoby znalezione przy innych wyborach w procesie przeszukiwania. Wybory, które nie zostały podjęte nazywane są rozbieżnościami (niezgodnościami). W konsekwencji, przez zwiększanie liczby dozwolonych niezgodności, strategia SBS systematycznie bada drzewo poszukiwań. Na wstępie dozwolona jest niewielka liczba niezgodności. Jeśli poszukiwanie nie jest zakończone sukcesem, wówczas liczba niezgodności zwiększa się aż do momentu, gdy znalezione zostanie rozwiązanie, bądź też przegląd drzewa zostanie wyczerpany.

Istnieje wiele odmian tego typu strategii, wśród nich najbardziej popularne to: strategia przeszukiwania głęboko-ograniczonej niezgodności (ang. Depth-Bounded Discrepancy Search, DDS), strategia przeszukiwania pierwszy-najlepszy (ang. Best-First Search, BFSearch), strategia przeszukiwania zagnieżdżonego (ang. Nested Search, NS), itp. [81]. Do drugiej grupy strategii można zaliczyć strategie określające sposób budowania drzewa przestrzeni potencjalnych rozwiązań. Na uwagę zasługują w tym przypadku strategie określające kolejność podstawiania zmiennych. Jedna z tego typu strategii [29], [32], [31], określa kolejność według rozmiaru dziedzin tych zmiennych. Przyjmowane jest, że w pierwszej kolejności pod uwagę brane są zmienne o najmniejszej dziedzinie. W pracach [53], [55], przedstawiona została uogólniona postać tej strategii. Dedykowana jest ona dla problemów z dynamicznie zmieniającą się, w trakcie przeszukiwania przestrzeni, liczbą zmiennych.

Z przedstawionej analizy widać, że projektant systemu wspomagania decyzji ma możliwość wyboru spośród szerokiego wachlarza rozwiązań. Istotna jest w tym przypadku znajomość problemu i doboru strategii pod kątem jego cech i właściwości.

W kontekście wyznaczania warunków wystarczających, dostępne strategie opierają się na przeglądzie zupełnym, bazującym na analizie tablicy prawdy. Tego typu strategia została wykorzystana do rozwiązania przykładów 2.4 i 2.5. Postępowanie to nie pozwala jednak na rozwiązywanie (w trybie interakcyjnym) problemów o większej (występującej w praktyce) liczbie formuł elementarnych. Do rozwiązania problemu decyzyjnego wykorzystuje się strategie, których istotą jest dekompozycja problemu na szereg podproblemów i rozwiązywanie ich sekwencyjnie lub równolegle [38].

Ze względu na charakter procedur wnioskowania metody logiczno-algebraicznej do rozwiązania problemu decyzyjnego można wykorzystać techniki programowania

z ograniczeniami. Układy równań (3.5) i (3.6) można wyrazić jako zbiór ograniczeń i poszukiwać w przestrzeni potencjalnych rozwiązań tych, które spełniają tak zdefiniowany zbiór ograniczeń.

4.3.1. Strategia przeszukiwania dwuetapowego

Rozwiązanie problemu decyzyjnego wymaga wyznaczenia wszystkich istniejących elementów zbioru S_u. Istotne jest, to że w przypadku wyznaczania warunków wystarczających, (faktów $Fu(u)$), wymagane jest posiadanie wiedzy tylko na temat wartości jakie przyjmują zmienne u. Zatem do wyznaczenia zbiorów S_{u1} (lub S_{u2}) konieczna jest znajomość wartości zmiennych u oraz gwarancja, że dla wartości zmiennych u istnieje co najmniej jedna kombinacja wartości zmiennych w i y, które spełniają narzucone ograniczenia. Stąd też nie jest niezbędne wyznaczanie wszystkich wartości zmiennych decyzyjnych (tak jak to zwykle jest realizowane w przypadku przeglądu zupełnego), a jedynie tylko wartości zmiennych u.

W oparciu o przedstawioną ideę opracowana została dwuetapowa strategia przeszukiwania zwana strategią przeszukiwania rozwiązań dedykowanych. Poniżej przedstawiono porównanie (ocenę) opracowanej strategii w stosunku do podejścia przeszukiwania całego drzewa przestrzeni potencjalnych rozwiązań.

Rozważmy w pierwszej kolejności rozwiązanie problemu decyzyjnego, przy użyciu technik programowania z ograniczeniami, polegające na dystrybucji wszystkich zmiennych decyzyjnych (jest to typowe podejście wykorzystywane w technikach programowania z ograniczeniami).

Rozwiązanie problemu decyzyjnego polega na wyznaczeniu wszystkich warunków wystarczających $Fu(u)$, które gwarantują spełnienie zadanej właściwości $Fy(y)$. Wymagane jest zatem wyznaczenie wszystkich rozwiązań dopuszczalnych problemów PSO_{Su1} i PSO_{Su2}. W klasycznym podejściu sprowadza się to do przeszukania całego drzewa przestrzeni potencjalnych rozwiązań, tzn. analizy kolejno wszystkich możliwy kombinacji zmiennych u, w, y. W celu zilustrowania jak pracochłonne (w sensie koniecznej liczby kroków obliczeniowych) jest takie postępowanie, rozważono poniższy przykład.

Przykład 4.8. Oszacowanie liczby kroków obliczeniowych dla PSO z trzema zmiennymi decyzyjnymi

Celem przykładu jest oszacowanie liczby kroków obliczeniowych koniecznych do przeszukania drzewa potencjalnych rozwiązań w przypadku gdy przeszukiwaniu podlega całe drzewo potencjalnych rozwiązań.

Dany jest problem: $PSO = (\{u, w, y\}, D), C)$,

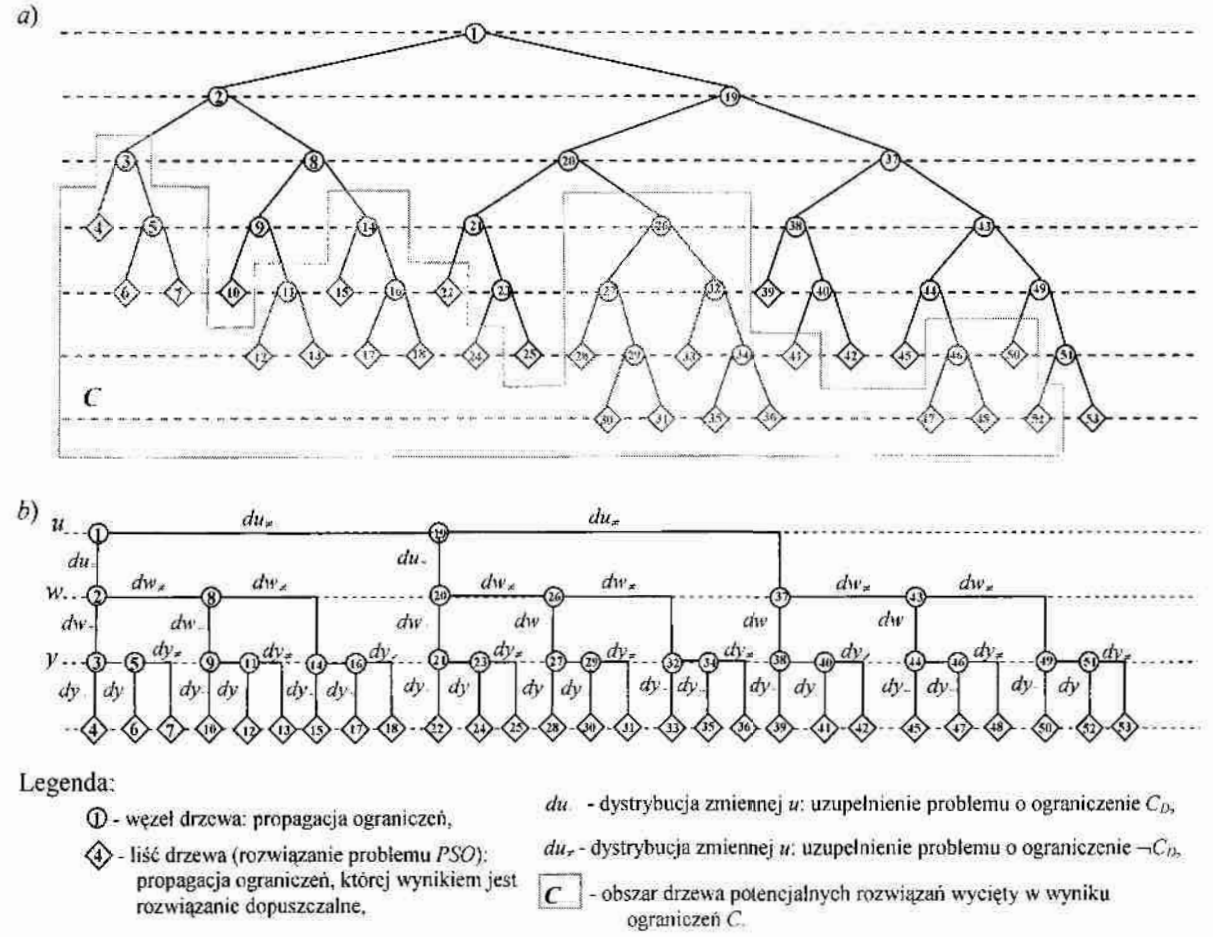
gdzie: u, w, y – zmienne decyzyjne, kolejno oznaczające zmienne wejściowe, pomocnicze i wyjściowe reprezentacji wiedzy KB,

$D = \{D_u, D_w, D_y\}$ – zbiór dziedzin zmiennych u, w, y, przyjęto że $||D_u|| = n_u = 3$, $||D_w|| = n_w = 3$, $||D_y|| = n_y = 3$,

C – zbiór ograniczeń opisanych na zbiorze zmiennych decyzyjnych u, w, y.

W rozważanym problemie wyróżniono trzy zmienne decyzyjne u, w, y, które mogą przyjmować wartości z trójelementowych dziedzin D_u, D_w, D_y. Poszukiwanie wartości zmiennych, dla których spełnione są ograniczenia C, polega na eksploracji drzewa potencjalnych rozwiązań przy wykorzystaniu mechanizmów propagacji ograniczeń dystrybucji zmiennych. W przykładzie przyjęto, że na każdym etapie dystrybucji problem jest dzielony na dwa podproblemy uzupełniane o ograniczenia dystrybucyjne postaci: C_D: $v = d_s$, $\neg C_D$: $v \neq d_s$, gdzie: v – jest jedną ze zmienny decyzyjnych u, w, y, podlegająca dystrybucji, d_s jest skrajną (maksymalną lub minimalną) wartością dziedziny D_v.

W pierwszej kolejności należy określić rozmiar drzewa potencjalnych rozwiązań R_T liczony jako suma węzłów i liści wchodzących w skład drzewa. Na rysunku 4.23 a) przedstawione zostało drzewo odpowiadające rozważanemu przykładowi. Rozmiar drzewa R_T = 53, zatem w przypadku najbardziej niekorzystnym (przyjęte ograniczenia nie zawężają rozmiaru drzewa) do wyznaczenia wszystkich rozwiązań dopuszczalnych należy wykonać 53 kroki obliczeniowe (pojedyncze węzły i liście drzewa oznaczają realizację pojedynczego kroku obliczeniowego).



Rys. 4.23. Drzewo potencjalnych rozwiązań: a) z zaznaczonym obszarem wyciętym w wyniku ograniczeń, b) w układzie poziomów dystrybucji jednej zmiennej

Na rysunku 4.23 b) przedstawiono drzewo z rysunku 4.23 a) w układzie czteropoziomowym, gdzie każdy poziom ilustruje procesy dystrybucji jednej określonej zmiennej. Na przykład, gałęzie pierwszego poziomu (pionowe i poziome) wychodzące

z węzłów 1 i 19 określają proces dystrybucji zmiennej u, z kolei gałęzie wychodzące z węzłów 2, 8, 20, 26, 37, 43, określają procesy dystrybucji dla zmiennej w. Gałęzie poziome określają etapy dystrybucji związane z dodawaniem ograniczeń typu $\neg C_D$, gałęzie pionowe określają etapy dystrybucji związane z dodawaniem ograniczeń typu C_D. Łatwo dostrzec związki między rozmiarem drzewa, a liczbą zmiennych i ich dziedzin. Liczba poziomów drzewa L_P jest o jeden większa od liczby zmiennych: $L_P = k_u + k_w + k_y$. Szerokość drzewa jest z kolei zależna do dziedzin poszczególnych zmiennych. Liczba węzłów L_{Ni} na i-tym poziomie jest krotnością, pomniejszonego o jeden, rozmiaru dziedziny zmiennej odpowiadającej danemu poziomowi: $L_{Ni} = j(n_i - 1)$, j – współczynnik krotności.
Rozmiar drzewa zgodnie z rysunkiem 4.23 b) jest obliczany jako suma węzłów każdego poziomu:

$$R_T = L_{N1} + L_{N2} + L_{N3} + L_{N4}.$$

Z kolei, liczbę węzłów dla kolejnych poziomów zgodnie z rysunkiem 4.23 b) wyznacza się z zależności:

$$L_{N1} = n_u - 1;\ L_{N2} = (n_w - 1)n_u;\ L_{N3} = (n_y - 1)\, n_w n_u;\ L_{N4} = n_y n_w n_u.$$

Stąd:

$$R_T = n_u - 1 + (n_w - 1)n_u + (n_y - 1)\, n_w n_u + n_y n_w n_u = 2n_y n_w n_u - 1. \qquad (4.19)$$

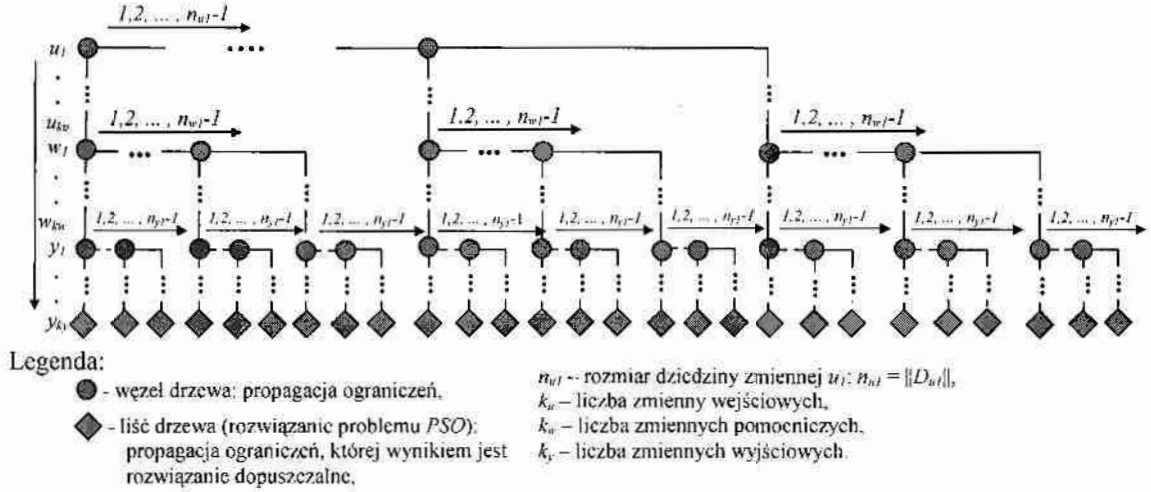
Z powyższej zależność można skorzystać przy wyznaczaniu rozmiaru drzewa potencjalnych rozwiązań. Dla danych przyjętych w przykładzie, rozmiar drzewa wynikający z przedstawionej zależności (4.19) wynosi 53.

W rzeczywistych przypadkach bardzo rzadko ma się do czynienia z sytuacją gdy konieczne jest przeszukiwanie całego drzewa. W wyniku wielokrotnej propagacji ograniczeń cześć gałęzi drzewa jest „odcinana". Na rysunku 4.23 a) szarym kolorem oznaczono przykładowy obszar, który został wycięty w wyniku propagacji ograniczeń. Stanowi on tę część drzewa, dla której wartości zmiennych decyzyjnych nie spełniają co najmniej jednego ograniczenia z zadanego zbioru C. W takim przypadku do wyznaczenia wszystkich rozwiązań dopuszczalnych wymagana jest realizacja 21 kroków obliczeniowych. Można stwierdzić, że liczba kroków obliczeniowych Z koniecznych do rozwiązania problemu *PSO* jest opisana następującą zależnością:

$$Z = \varepsilon(C) \cdot R_T. \qquad (4.20)$$

W przedstawionym wyrażeniu, funkcja $\varepsilon(C)$ określa w jakim stopniu rozmiar drzewa potencjalnych rozwiązań zostaje zawężony w wyniku zadanego zbioru ograniczeń C: $\varepsilon(C)$: $C \rightarrow [0,1]$. W wielu praktycznych przypadkach *PSO* bardzo trudno (jeśli w ogóle to możliwe) jest określić postać funkcji $\varepsilon(C)$. Zwykle też liczba kroków obliczeniowych Z, koniecznych do rozwiązania problemu *PSO*, wyznaczana jest na drodze eksperymentalnej.

W przykładzie 4.8. przedstawiono koncepcję wyznaczania liczby kroków obliczeniowych koniecznych do wyznaczenia wszystkich rozwiązań problemu *PSO* opisanego przez trzy zmienne decyzyjne. Problemy PSO_{Su1} i PSO_{Su2} (wykorzystywane do rozwiązania problemu decyzyjnego) są opisywane przez większą liczbę zmiennych decyzyjnych (wchodzących w składy sekwencji *u*, *w*, *y*), gdzie każda z nich może być opisana przez inną dziedzinę. Drzewo potencjalnych rozwiązań, w układzie poziomów reprezentujących dystrybucję jednej zmiennej przedstawione zostało na rysunku 4.24.



Rys. 4.24. Ogólna postać drzewa potencjalnych rozwiązań w układzie poziomów dystrybucji jednej zmiennej dla problemów PSO_{Su1} i PSO_{Su2}

Postępując analogicznie jak w przykładzie 4.8. wyznaczono zależność opisującą rozmiar R_{TI} drzewa:

$$R_{TI} = 2(n_u^{ku} \cdot n_w^{k_w} \cdot n_y^{k_y}) - 1, \tag{4.21}$$

gdzie: n_u, n_w, n_y, – określają kolejno liczbę elementów zbiorów D_u, D_w, D_y: $n_u = ||D_u||$, $n_w = ||D_w||$, $n_y = ||D_y||$,

k_u, k_w, k_y – określają kolejno rozmiar sekwencji *u*, *w*, *y*: $k_u = ||u||$, $k_w = ||w||$, $k_y = ||y||$.

Liczba kroków konieczna do wyznaczania wszystkich rozwiązań dopuszczalnych problemu PSO_{Su1} (lub PSO_{Su1}) zgodnie z zależnością (4.20) ma postać:

$$Z_I = \varepsilon_I(C) \cdot \left[2(n_u^{ku} \cdot n_w^{k_w} \cdot n_y^{k_y}) - 1\right], \tag{4.22}$$

Należy podkreślić, że przedstawione wyrażenie może być wykorzystywane do wyznaczania liczby kroków obliczeniowych w problemach, w których mechanizm dystrybucji zmiennych polega na podziale problemu na dwa podproblemy i uzupełnienie ich dwa przeciwne ograniczenia dystrybucyjne C_D, $\neg C_D$. Jest to najczęściej używany mechanizm dystrybucji w komercyjnych środowiskach programowania z ograniczeniami [77], [99].

Analogiczne wyrażenia można budować dla innych mechanizmów dystrybucji, np. polegających na podziale problemu na trzy podproblemy.

Wyrażenie 4.22 jest iloczynem dwóch elementów (funkcji $\varepsilon_I(C)$ i rozmiaru drzewa $R_{TI} = 2(n_u^{ku} \cdot n_u^{ku} \cdot n_u^{ku})$-1), które w sposób przeciwstawny wpływają na liczbę kroków obliczeniowych. Wraz ze wzrostem rozmiaru problemu (rosnący rozmiar dziedzin, rosnąca liczba zmiennych), rozmiar drzewa potencjalnych R_{TI} rozwiązań rośnie wykładniczo. Wraz ze wzrostem liczby ograniczeń maleje wartość funkcji $\varepsilon_I(C)$ co prowadzi do zmniejszana liczby kroków obliczeniowych. W wielu przypadkach wpływ funkcji $\varepsilon_I(C)$ jest na tyle niewielki, że otrzymana liczba kroków obliczeniowych jest na tyle duża, że niemożliwe okazuje się przeszukanie całego drzewa potencjalnych rozwiązań w czasie akceptowalnym przez użytkownika.

Na wstępie rozdziału przedstawiona została właściwość polegająca na tym, że w przypadku rozwiązywania problemów decyzyjnych nieistotne są wartości zmiennych w i y. Istotne jest posiadanie wiedzy na temat zmiennych u gwarantujących, że istnieje co najmniej jedna kombinacja zmiennych w i y spełniających zadane ograniczenia.
Własność ta umożliwia świadome pomijanie podczas przeszukiwania pewnych obszarów (gałęzi związanych ze zmiennymi w i y) drzewa potencjalnych rozwiązań. Ograniczenie obszaru przeszukiwania przyczynia się do ograniczenia niezbędnej liczby kroków obliczeniowych. Przedstawiona własność została wykorzystana do opracowania strategii efektywnego czasowo (w trybie *on-line*) poszukiwania rozwiązania tego problemu.

Zaproponowana strategia polega na przeszukiwaniu drzewa w dwóch etapach. Istota strategii sprowadza się do częściowej dystrybucji zmiennych decyzyjnych – gdzie przez częściową dystrybucję rozumie się przeszukiwanie drzewa przy wykorzystaniu dystrybucji tylko dla wybranej grupy zmiennych (a nie jak dotychczas dystrybucji wszystkich zmiennych).

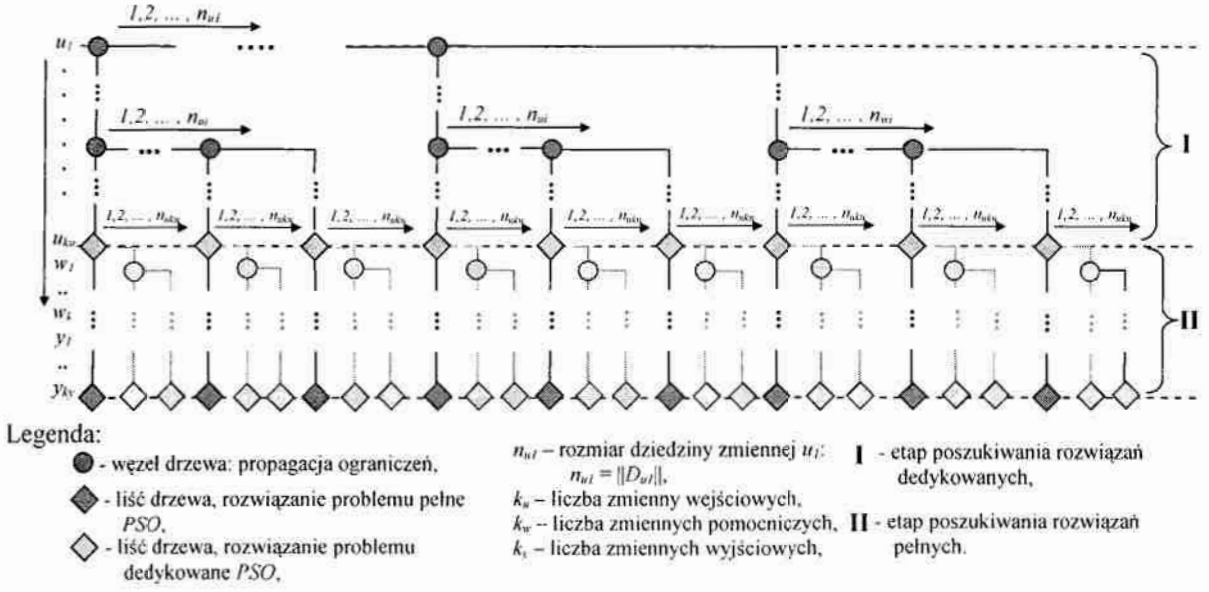
Na użytek dalszych rozważań wprowadza się pojęcia rozwiązań pełnych i dedykowanych. Przez rozwiązanie dedykowane rozumiana jest trójka (u, w, y) (gdzie u, w, y - oznaczają kolejno zmienne wejściowe, wewnętrzne, wyjściowe bazy wiedzy), w której wszystkie elementy u, posiadają jednoelementowe dziedziny: $\forall D_{ui} \in D : \|D_{ui}\| = 1$, a dziedziny elementów sekwencji w, y pozostają wieloelementowe: $(\exists D_{wi} \in D : \|D_{wi}\| > 1) \vee$ $\vee (\exists D_{wi} \in D : \|D_{wi}\| > 1)$.

Przez rozwiązanie pełne rozumiana jest taka postać zmiennych u, w, y, w których wszystkie elementy posiadają dziedziny jednoelementowe.
W strategii przeszukiwania wyróżnia się dwa etapy postępowania, rysunek 4.25:

- poszukiwanie rozwiązań dedykowanych (z wieloelementowymi dziedzinami),
- sprawdzanie istnienia rozwiązań pełnych.

W pierwszym etapie poszukiwane są rozwiązania dedykowane (dystrybucja tylko wartości sekwencji u). Wyznaczone w ten sposób rozwiązania charakteryzują się tym, że tylko elementy wektora u posiadają określoną wartość, tzn. tylko dla elementów sekwencji u dziedziny są jednoelementowe.

W drugim etapie sprawdzane jest istnienie rozwiązań pełnych. Polega ono na tym, że dla każdego z wyznaczonych rozwiązań dedykowanych przeprowadzane jest poszukiwanie (poprzez dystrybucję zmiennych w, y), tylko jednego rozwiązania (jednej gałęzi drzewa). Znalezienie choć jednego rozwiązania pełnego jest dowodem na to, że dla danego rozwiązania dedykowanego (czyli dla pewnych wartości u) istnieją wartości w, y, spełniające ograniczenia $Q(u, w, y) = \bar{1}$, $Qy(y) = \bar{1}$.



Rys. 4.25. Ogólna postać drzewa potencjalnych rozwiązań w układzie poziomów dystrybucji jednej zmiennej dla dwuetapowej strategii przeszukiwania rozwiązań dedykowanych

Zaproponowana strategia pozwala na znaczne ograniczenie liczby kroków obliczeniowych. Dzięki temu, że w drugim etapie dla każdego rozwiązania dedykowanego poszukiwane jest tylko jedno rozwiązanie pełne, w obszarze drzewa odpowiadającym zmiennym w i y wyznaczana jest tylko jedna gałąź dopuszczalna, pozostałe gałęzie są ignorowane (kolor szary). Nie jest konieczne tym samym przeszukiwanie całego drzewa potencjalnych rozwiązań lecz tylko jego pewnego fragmentu. Rozmiar przeszukiwanego drzewa R_{T2} można wyznaczyć z zależności:

$$R_{T2} = R_{T2I} + R_{T2II}, \tag{4.23}$$

gdzie: R_{T2I}, R_{T2II}, określają odpowiednio liczbę kroków obliczeniowych w etapie **I** i etapie **II.**

Liczba kroków obliczeniowych w etapie pierwszym R_{T2I} jest wyznaczana z zależności (4.21) przy założeniu, że istnieje tylko sekwencja u. Wielkość R_{T2I} opisana jest zależnością:

$$R_{T2I} = 2n_u^{ku} - 1, \tag{4.24}$$

gdzie: n_u – określa liczbę elementów zbioru D_u: $n_u = ||D_u||$,
k_u– określają rozmiar sekwencji u: $k_u = ||u||$,

Ze względu na to, że w etapie drugim poszukiwane jest tylko jedno rozwiązanie (pojedyncza gałąź), liczba węzłów jest krotnością sumy liczby elementów sekwencji w i y: $R_{T2II} = j \cdot (k_w + k_y)$, gdzie suma $k_w + k_y$ oznacza liczbę węzłów jednej gałęzi, j - jest liczbą gałęzi równą liczbie rozwiązań dedykowanych. W najgorszym przypadku, j jest równe maksymalnej liczbie rozwiązań dedykowanych $j = n_u^{ku}$, R_{T2II} jest postaci:

$$R_{T2II} = n_u^{ku} \cdot (k_w + k_y), \tag{4.25}$$

gdzie: n_u – określa liczbę elementów zbioru D_u: $n_u = \|D_u\|$,

k_u– określają rozmiar sekwencji u: $k_u = \|u\|$,

k_u, k_w, k_y – określają kolejno rozmiar sekwencji u, w, y: $k_u = \|u\|$, $k_w = \|w\|$, $k_y = \|y\|$.

Z zależności (4.23), (4.24) wynika, że rozmiar drzewa potencjalnych rozwiązań (przy zastosowaniu dwuetapowej strategii przeszukiwania ma postać)

$$R_{T2} = n_u^{ku} (2 + k_w + k_y) - 1. \tag{4.26}$$

Zgodnie z zależnością (4.20) liczba kroków obliczeniowych opisuje zależność:

$$Z_2 = \varepsilon_2(C) \cdot \left[n_u^{ku} (2 + k_w + k_y) - 1 \right]. \tag{4.27}$$

Podobnie jak w przypadku klasycznego podejścia, rozmiar drzewa podlegający przeszukiwaniu rośnie wykładniczo wraz ze zwiększaniem się rozmiaru problemu. Jednak w przypadku stosowania dwuetapowej strategii przeszukiwania obszar jest znacznie mniejszy niż w przypadku klasycznym. Zakładając dla uproszczenia, że $\varepsilon_1(C) = \varepsilon_2(C)$ (co nie zawsze jest prawdą) stosunek liczby kroków obliczeniowych w przypadku klasycznym i zastosowaniu dwuetapowej strategii przeszukiwania Z_2/Z_1 można wyrazić następująco:

$$\frac{Z_2}{Z_1} \approx \frac{2 + k_w + k_y}{2 n_w^{k_w} n_y^{k_y}}. \tag{4.28}$$

Stosunek Z_2/Z_1 stanowi oszacowanie zysku stosowania dwuetapowej strategii przeszukiwania rozwiązań dedykowanych w stosunku do podejścia opartego na całkowitej dystrybucji zmiennych.

Przykład 4.9. Oszacowanie zysku stosowania dwuetapowej strategii przeszukiwania

Przedstawiony przykład ilustruje oszacowanie zysku stosowania dwuetapowej strategii przeszukiwania rozwiązań dedykowanych dla zadanego problemu *PSO*.
Dany jest problem:

$$PSO_{Su1} = (V, D), C),$$

gdzie: $V = u \cup w \cup y$,

$u = \{u_1, u_2, u_3\}$, $w = \{w_1, w_2, w_3, w_4\}$, $y = \{y_1, y_2\}$,

$D = D_u \cup D_w \cup D_y$, $D_u = \{D_{u,1}, D_{u,2}, D_{u,3}\}$, $D_{u,i} = \{1, 2, 3, 4, 5\}$ dla $i = 1,2,3$, $D_w = \{D_{w,1}, D_{w,2}, D_{w,3}, D_{w,4}\}$, $D_{w,i} = \{1, 2, 3\}$ dla $i = 1, \ldots, 4\}$, $D_y = \{D_{y,1}, D_{y,2}\}$, $D_{y,i} = \{1, 2, \ldots, 7\}$ dla $i = 1,2\}$,

$C = \{Q(u, w, y) = \bar{1}, Qy(y) = \bar{1}\}$.

Dla tak zdefiniowanego problemu (odpowiadającego układowi (3.5)) poszukiwany jest zbiór S_{u1} wartości zmiennych wejściowych, dla których spełnione są ograniczenia C:

$$S_{u1} = \{u: Q(u, w, y) = \bar{1}, Qy(y) = \bar{1}\} .$$

Dla podanych danych rozmiary sekwencji i dziedzin wynoszą: $n_u = 5$, $n_w = 3$, $n_y = 7$, $k_u = 3$, $k_w = 4$, $k_y = 2$.

Zgodnie z zależnością (4.21) rozmiar drzewa potencjalnych rozwiązań w typowym podejściu opartym na całkowitej dystrybucji zmiennych wynosi: $R_{T1} = 2{,}6 \cdot 10^6$.

W przypadku zastosowania dwuetapowej strategii przeszukiwania rozmiar drzewa potencjalnych rozwiązań R_{T2}, zgodnie z zależnością (4.26), wynosi: $R_{T2} = 1000$.

Zysk stosowania dwuetapowej strategii przeszukiwania rozwiązań dedykowanych zgodnie z (4.28) wynosi:

$$\frac{Z_2}{Z_1} \approx 3{,}8 \cdot 10^{-4} .$$

W rozważanym przykładzie zysk z zastosowania zaproponowanej strategii pozwala uzyskać rozwiązanie przy zmniejszonym nakładzie obliczeniowym o 4 rzędy wielkości.

■

Zysk otrzymywany w przypadku stosowania dwuetapowej strategii przeszukiwania rozwiązań dedykowanych jest wynikiem wykorzystania dostrzeżonej właściwości (polegającej na tym, że dla problemu decyzyjnego istotne są tylko wartości zmiennych wejściowy u) . W tym kontekście zaproponowana strategia stanowi szczególny przypadek metody podziału i ograniczeń. W pierwszym etapie odbywa się (podczas cyklicznie powtarzanych procesów propagacji i dystrybucji) ocena potencjalnych rozwiązań pod kątem wartości elementów sekwencji wejściowej u, które spełniają zadane ograniczenia C. Inaczej mówiąc, w zbiorze wszystkich możliwych wartości odbywa się wstępna selekcja rozwiązań, które w kontekście określonych zmiennych spełniają wymagane ograniczenia.

Każdemu z otrzymanych rozwiązań dedykowanych odpowiada określona przestrzeń potencjalnych wartości elementów sekwencji w i y.

W drugim etapie, osobno dla każdego rozwiązania dedykowanego odbywa się ocena (poprzez przeszukiwanie przestrzeni potencjalnych wartości elementów sekwencji w i y) określająca, czy wartości elementów sekwencji w i y spełniają zadane ograniczenia. Taka weryfikacja pozwala stwierdzić, które z rozwiązań dedykowanych są poszukiwanymi rozwiązaniami pełnymi (dopuszczalnymi). Istotą przedstawionej strategii jest wstępna selekcja, która pozwala ograniczyć się w dalszych poszukiwaniach tylko do przestrzeni rozwiązań dedykowanych.

W dodatku C przedstawione zostały wyniki eksperymentów porównawczych między zaproponowaną strategią a metodami opartymi na przeglądzie zupełnym oraz algorytmami rekurencyjnymi. W obu przypadkach wyniki wskazywały na korzyść dwuetapowej strategii przeszukiwania.

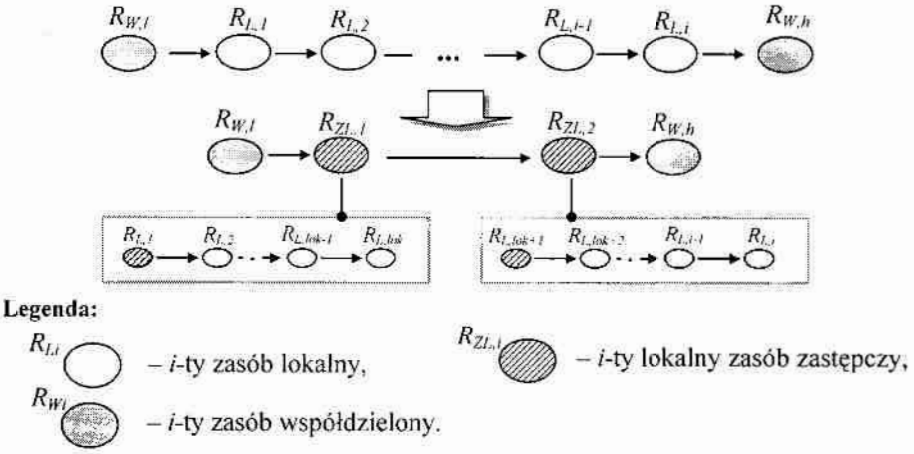
Dwuetapowa strategia przeszukiwania rozwiązań dedykowanych stanowi podstawowy mechanizm wykorzystywany w procesie rozwiązywania problemu decyzyjnego, a tym samym w procesie wyznaczania warunków wystarczających i weryfikacji bazy wiedzy. Poza przedstawioną strategią, w celu zmniejszenia koniecznej liczby kroków obliczeniowych, wykorzystywane są również mechanizmy minimalizujące liczbę zmiennych decyzyjnych (przez co zmniejsza się rozmiar przestrzeni potencjalnych rozwiązań). W przypadku systemów transportowych odbywa się to poprzez kompresję struktury połączeń między zasobami. W wyniku kompresji struktury systemu, reprezentacja wiedzy *KB* charakteryzowana jest przez mniejszą liczbę faktów co w efekcie prowadzi do mniejszej liczby ograniczeń i mniejszej liczby zmiennych decyzyjnych.

4.3.2. Kompresja struktury systemu współbieżnych procesów cyklicznych

Zaproponowana w punkcie 4.2 struktura schematu faktów $F_{Pa}(\Theta, S_0, x, P_D, pr)$ umożliwia automatyczne generowanie reprezentacji wiedzy stanowiących opis ogólnych zasad funkcjonowania systemu. Systemy transportowe o rozmiarach spotykanych w rzeczywistości (typowe wielkości to ok. 20 wózków samojezdnych i 50 sektorów) opisywane są przez reprezentację wiedzy (*KB*) w skład, której wchodzi około 1000 faktów i 200 zmiennych decyzyjnych.

Poszukiwanie warunków wystarczających poprzez rozwiązanie problemów PSO przy tak dużej liczbie zmiennych, nawet przy wykorzystaniu dwuetapowej strategii przeszukiwania rozwiązań dedykowanych, związane jest z dużym nakładem czasowym, którym nie zawsze dysponuje użytkownik systemu. Prezentowane poniżej rozwiązania polegają na zmniejszeniu liczby faktów i zmiennych poprzez kompresję struktury systemu SWPC.

Pierwsze usprawnienie polega na „scalaniu" ze sobą zasobów lokalnych należących do wspólnego procesu. W tym celu, w strukturze SWPC wprowadza się pojęcie gałęzi. Gałęzią struktury SWPC, nazywany jest zbiór zasobów $R_g \subset R$, połączonych ze sobą szeregowo i charakteryzujących się tym samym zbiorem obsługiwanych procesów. Jeżeli i zasobów lokalnych $R_{L,1}, R_{L,2}, \ldots, R_{L,i}$, gdzie: $R_{L,1}, R_{L,2}, \ldots, R_{L,i} \in R$, tworzy gałąź w strukturze SWPC, to gałąź ta może być reprezentowana przez dwa zasoby zastępcze $R_{ZL,1}$ i $R_{ZL,2}$, gdzie: $R_{ZL,1} \cup R_{ZL,2} = \{R_{L,1}, R_{L,2}, \ldots, R_{L,i}\}$, $R_{ZL,1} = \{R_{L,1}, \ldots, R_{L,lok}\}$, $R_{ZL,1} = \{R_{L,lok+1}, \ldots, R_{L,i}\}$, $lok \in \{1, 2, \ldots, i\text{-}1\}$, (rysunek 4.26).



Rys. 4.26. Idea scalania zasobów lokalnych

Tak definiowane zasoby $R_{ZL,1}$, $R_{ZL,2}$ tworzą grupy zasobów lokalnych będących częścią gałęzi. Zasoby zastępcze traktowane są jak typowe zasoby lokalne. Obowiązują dla nich takie same zasady i ograniczenia jak w przypadku typowego zasobu lokalnego. Przypisanie do procesu P_i zasobu $R_{ZL,1}$ (lub $R_{ZL,2}$) oznacza, że proces jest realizowany kolejno na zasobach $R_{L,1}$, ..., $R_{L,lok1}$. Przypisanie zasobu $R_{ZL,1}$ (lub $R_{ZL,2}$) do sekwencji S_0 oznacza, że określony proces P_i (realizujący operacje na zasobach $R_{L,1}$, $R_{L,2}$, ..., $R_{L,i}$) rozpoczyna swoją operację od pierwszego zasobu reprezentowanej grupy czyli od $R_{L,1}$ (lub w przypadku $R_{ZL,2}$ od zasobu $R_{L,lok+1}$). Jeżeli proces kończy swoją pracę na zasobie $R_{ZL,1}$ (lub $R_{ZL,2}$) oznacza to, że proces kończy swoją pracę na ostatnim zasobie grupy $R_{ZL,1}$ (lub $R_{ZL,2}$).

Przyporządkowanie zasobów lokalnych do zasobów zastępczych może odbywać się na wiele sposobów. W zależności od wartości parametru *lok* poszczególne zasoby lokalne mogą raz należeć do zbioru $R_{ZL,1}$ innym razem zaś do zbioru $R_{ZL,2}$: np. dla *lok* = 1 zbiory zastępcze mają postać: $R_{ZL,1} = \{R_{L,1}\}$, $R_{ZL,2} = \{R_{L,1}, R_{L,2}, \ldots, R_{L,i}\}$. Innymi słowy, dla zasobów zastępczych istnieje i-1 (i-liczba zasobów lokalnych) możliwych postaci. Posługiwanie się zasobami zastępczymi pozwala na budowanie reprezentacji wiedzy *KB* z mniejszą liczbą zmiennych (a tym samym mniejszą liczbą faktów) oraz możliwość wyznaczania ogólnej postaci warunków wystarczających (to znaczy warunków w postaci stanu początkowego i reguł priorytetowania, którym odpowiada więcej niż jeden harmonogram pracy).

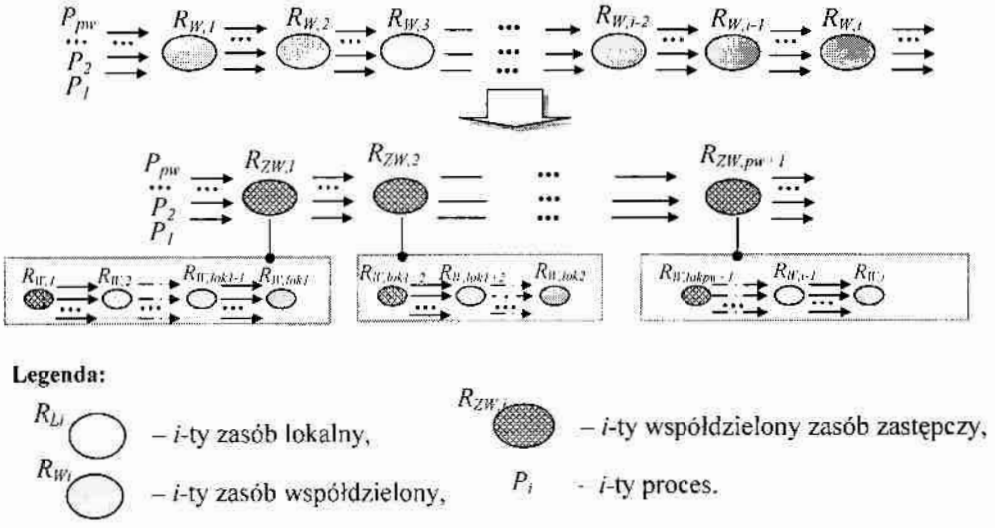
W punkcie 4.6 stwierdzono, że jednej postaci sekwencji S_0 i Θ odpowiada dokładnie jedna sekwencja x. Stosowanie zasobów zastępczych umożliwia wyznaczanie sekwencji S_0 i Θ, którym odpowiadać będzie wiele sekwencji x (harmonogramów). Dla przykładu, sekwencja $S_0 = (R_{ZL,2}, R_2)$ oznacza, że w określonym systemie proces P_1 rozpoczyna się od zasobu $R_{ZL,2}$. Łatwo wykazać, że w zależności od wartości zmiennej *lok*, zasób $R_{ZL,2}$ może przyjąć jedną z i-1 postaci, a zatem sekwencji S_0 odpowiada i-1 harmonogramów.

Kolejne usprawnienia opierają się na analogicznej idei jednak dotyczą kompresji zasobów współdzielonych. Wyróżnia się kompresję zasobów współdzielonych dla procesów jednokierunkowych i kompresję dla procesów dwukierunkowych.

Kompresja dla procesów jednokierunkowych została zilustrowana na rysunku 4.27. Usprawnienie to polega na „scalaniu" ze sobą zasobów współdzielonych tworzących gałąź,

w której realizowany jest ten sam zbiór procesów. Wszystkie procesy są realizowane w jednym kierunku. Jeśli i zasobów współdzielonych $R_{W,1}$, $R_{W,2}$, ..., $R_{W,i}$, gdzie: $R_{W,1}, R_{W,2}, \ldots, R_{W,i} \in R$, tworzy gałąź w strukturze SWPC, to gałąź ta może być reprezentowana przez zasoby zastępcze $R_{ZW,1}$, $R_{ZW,2}$, ... , $R_{ZW,pw+1}$, gdzie: pw – oznacza liczbę procesów realizowanych na zasobach; $R_{ZW,1} \cup R_{ZW,2} \cup , \ldots , \cup R_{ZW,pw+1} = \{R_{W,1}, R_{W,2}, \ldots , R_{W,i}\}$, $R_{ZW,1} = \{R_{W,1}, \ldots , R_{W,lok1}\}$, $R_{ZW,2} = \{R_{W,lok1+1}, \ldots , R_{W,lok2}\}$, $R_{ZW,3} = \{R_{W,lok2+1}, \ldots , R_{W,lok3}\}$, ... , $R_{ZW,pw+1} = \{R_{W,lokpw+1}, \ldots , R_{W,i}\}$, $lok_j \in \{j, 2, \ldots , i\text{-}1\text{-}pw+j\}$, $lok_1 < lok_2 < \ldots < lok_{pw}$.

Zasoby zastępcze $R_{ZW,1}$, $R_{ZW,2}$, ... , $R_{ZW,pw+1}$, traktowane są jak typowe zasoby współdzielone. Obowiązują dla nich takie same zasady i ograniczenia jak w przypadku typowego zasobu współdzielonego. Dla każdego zasobu $R_{ZW,i}$ zastępczego przyporządkowywana jest reguła priorytetowania $\sigma_{ZW,i}$, która określa kolejność obsługi procesów na zasobie zastępczym. Przyporządkowanie reguły $\sigma_{ZW,i}$ do zasobu zastępczego jest równoznaczne z przydzieleniem takich samych reguł do wszystkich zasobów R_W reprezentowanych przez zasób zastępczy. Inaczej mówiąc, kolejność obsługi procesów przez zastępcze zasoby współdzielone należące do zbioru $R_{ZW,i}$ opisuje jedna wspólna reguła priorytetowania. Przypisanie zasobu $R_{ZW,i}$ do sekwencji S_0 oznacza, że określony proces P_j (realizujący operacje na zasobach $R_{W,1}$, $R_{W,2}$, ..., $R_{W,q}$) rozpoczyna swoją operację od pierwszego zasobu reprezentowanej grupy, czyli od $R_{W,lok(i-1)+1}$. W przypadku gdy proces P_j kończy operację na zasobie zastępczym $R_{ZW,i}$ oznacza to w praktyce, że kończy swoją pracę na ostatnim zasobie współdzielonym wchodzącym w skład $R_{ZW,i}$ (kończy pracę na zasobie $R_{W,loki}$).

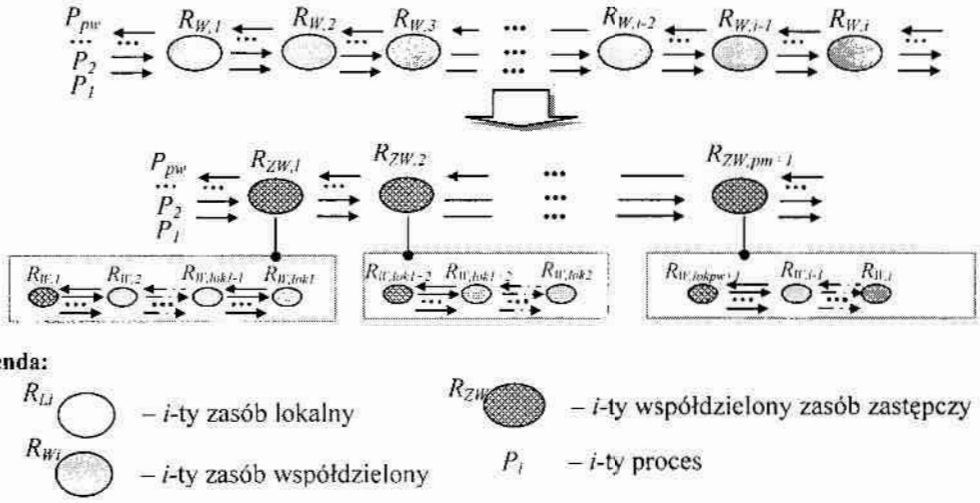


Rys. 4.27. Idea scalania zasobów współdzielonych

Podobnie jak w przypadku zasobów lokalnych zastępczy zasób współdzielony $R_{ZW,i}$ odpowiada pierwszemu zasobowi reprezentowanego zbioru. Przyjęta liczba zasobów zastępczych wynosi $pw + 1$. Liczba ta wynika z złożenia, że w gałęzi powinno być tyle zasobów by każdy z realizowanych procesów mógł się w niej ukończyć oraz jeden zasób został jeszcze wolny. Zmienne lok_j określają granice przydziału zasobów do zbiorów $R_{ZW,j}$.

Zatem w zależności od wartości zmiennych lok_j zbiory $R_{ZW,j}$ mogą przyjmować różne postacie. Przyjmując, że liczba zasobów współdzielonych wynosi kw, liczba procesów pw, to liczba możliwości przyporządkowania zasobów współdzielonych do zasobów zastępczych $R_{ZW,j}$ jest równa $\frac{(kw-1)!}{(kw-1-pw)!\cdot pw!}$.

Analogicznie do przedstawionej idei odbywa się „scalanie" zasobów współdzielonych w przypadku dwukierunkowej realizacji procesów. Idea ta została przedstawiona na rysunku 4.28. Gałąź, w skład której wchodzą zasoby współdzielone $R_{W,1}$, $R_{W,2}$, ..., $R_{W,i}$, podobnie jak poprzednio, reprezentowana jest przez zasoby zastępcze $R_{ZW,1}$, $R_{ZW,2}$, ..., $R_{ZW,pm+1}$.



Rys. 4.28. Idea scalania zasobów współdzielonych przy realizacji procesów w dwóch kierunkach

Jednak w tym przypadku liczba zasobów zastępczych jest mniejsza niż w przypadku gałęzi jednokierunkowych. Liczba zasobów zastępczych jest określana jako maksymalna wartość z liczby procesów realizowanych w różnych kierunkach: $pm = \max\{pwl, pwp\}$, gdzie: pwl – liczba procesów realizowanych w kierunku lewym, pwp – liczba procesów realizowanych w kierunku prawym. Przyjęta liczba zasobów zastępczych jest wynikiem założenia, że w danej gałęzi ukończyć mogą pracę tylko procesy realizowane w jednym kierunku. Zatem liczba zasobów zastępczych odpowiada liczbie procesów realizowanych w jednym kierunku plus jeden zasób wolny.

Celem wprowadzenia przedstawionych usprawnień jest minimalizacja liczby zmiennych i faktów wchodzących w skład reprezentacji wiedzy *KB*. W wyniku kompresji otrzymywane są mniejsze struktury, a co za tym idzie do opisu określonego systemu transportowego wymagana jest mniejsza liczba faktów i zmiennych decyzyjnych.

Należy jednak podkreślić, że prezentowane usprawnienia mogą być wykorzystywane jedynie w przypadkach gdy w trakcie realizacji procesów nie są istotne czasy trwania poszczególnych operacji (lub przyjmowane są czasy jednostkowe operacji).

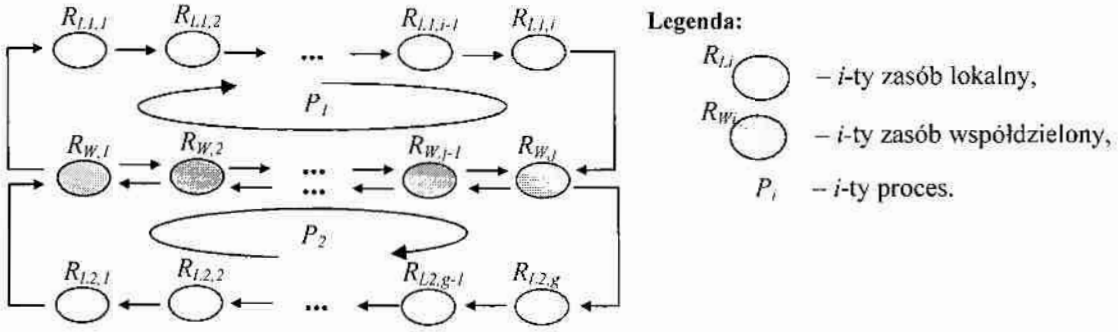
Inaczej mówiąc kompresja faktów ma sens w przypadku poszukiwania rozwiązań w dziedzinie stanów. Poszukiwanie rozwiązań w dziedzinie stanów polega na rozstrzygnięciu kolejności wykonywania poszczególnych operacji. Upływ czasu jest rozpatrywany od stanu do

stanu. W tym kontekście kompresja struktury systemu transportowego może być wykorzystywana tylko dla wyznaczania warunków wystarczających gwarantujących istnienie odpowiedzi na pytania określające właściwości wyjściowe w dziedzinie stanów.

Mimo tego ograniczenia, usprawnienia te są szczególnie przydatne przy wyznaczaniu warunków gwarantujących unikanie blokad. Poniższy przykład ilustruje zysk wynikający ze stosowania przedstawionych usprawnień.

Przykład 4.10. Kompresja *KB*

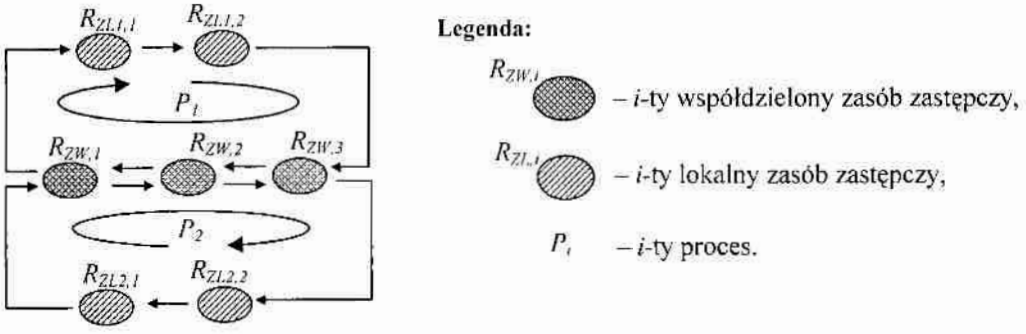
Celem przykładu jest ilustracja sposobu zmniejszenia liczby faktów reprezentacji wiedzy (*KB*) systemu SWPC z rysunku 4.29.



Rys. 4.29. Przykład systemu SWPC

W systemie z rysunku 4.29. realizowane są cyklicznie dwa procesy P_1 i P_2. W skład systemu wchodzą trzy gałęzie, dwie z zasobami lokalnymi $R_{L1,i}$, $R_{L2,g}$ i jedna tworzona przez zasoby współdzielone $R_{W,j}$. Wielkości i, g określają liczbę zasobów lokalnych, j oznacza liczbę zasobów współdzielonych. Zgodnie z przyjętą postacią schematu faktów $F_{Pd}(\Theta, S_0, x, P_D, pr)$, w skład reprezentacji wiedzy *KB*, opisującej system z rysunku 4.29 wchodzi $2+i+g+4j$ zmiennych (zmienne Θ, S_0, x) i $10j+i+g$ faktów. Przyjmując w uproszczeniu, że liczby zasobów w gałęziach są sobie równe: $i = j = g$, to liczba zmiennych wynosi $2+6i$, a liczba faktów $12i$.

System SWPC w wyniku zastosowania zaproponowanych usprawnień został skompresowany do postaci przedstawionej na rysunku 4.30.



Rys. 4.30. Przykład systemu SWPC po kompresji

W skład reprezentacji wiedzy *KB* opisującej system po kompresji wchodzi 14 zmiennych i 28 faktów, jest to liczba stała. Na przykład system z rysunku 4.29, dla którego liczba zasobów i w każdej gałęzi wynosi $i = 20$, jest opisany za pomocą 240 faktów. W wyniku kompresji liczba faktów zmniejsza się ponad 8 razy (zamiast 240 w wyniku kompresji jest 28 faktów). Stosowanie przedstawionych zasad umożliwia zatem reprezentację systemu SWPC przy użyciu mniejszej liczby faktów i zmiennych.

■

Przestawione podejście do kompresji liczby faktów i zmiennych decyzyjnych w połączeniu z dwuetapową strategią przeszukiwania rozwiązań dedykowanych znacznie ograniczają liczbę kroków obliczeniowych koniecznych do rozwiązywania problemu decyzyjnego. Szczególnie istotny zysk otrzymywany jest w przypadkach gdy wnioskowanie odbywa się dla faktów wyjściowych o charakterze stanowym.

W kolejnym rozdziale przedstawiono procedury wyznaczania warunków wystarczających dla systemów transportowych, dla których poszukiwane są rozwiązania bezblokadowe i bezkolizyjne. Procedury bazują na przedstawionych metodach kompresji i zaproponowanej strategii przeszukiwania. Proces wyznaczania warunków wystarczających stanowi ostatni etap procesu weryfikacji bazy wiedzy (punkt 4.1).

4.3.3. Procedury wyznaczania warunków wystarczających

Weryfikacja bazy wiedzy polega na wyznaczeniu warunków wystarczających (właściwości wejściowej $Fu(S_0,\Theta)$), dla których spełnione są żądania użytkownika definiowane w postaci właściwości wyjściowej $Fy(x)$. Warunki te mogą być poszukiwane dla dowolnych postaci właściwości wyjściowej $Fy(x)$. Aby możliwe było poszukiwanie tych warunków konieczne jest posiadanie reprezentacji, która umożliwi prowadzenie poprawnego wnioskowania. W punkcie 4.2.4 przedstawiony został schemat faktów umożliwiający wyznaczanie warunków gwarantujących istnienie (dla systemów transportowych rozważanej klasy) harmonogramów bezblokadowych i bezkolizyjnych.

Wymagane było, by dla zrealizowanego schematu faktów, słuszne było Twierdzenie 4.1. Zatem twierdzenie to stanowi niejako warunek budowy schematu faktów. W tym kontekście wyróżnić można dwa typy warunków wystarczających:

- warunki gwarantujące poprawny proces wnioskowania – wykorzystywane przy budowie schematu faktów (dla rozważanej klasy systemów transportowych jest to Twierdzenie 4.1),
- warunki w postaci właściwości wejściowych Fu gwarantujące spełnienie żądanych właściwości wyjściowych.

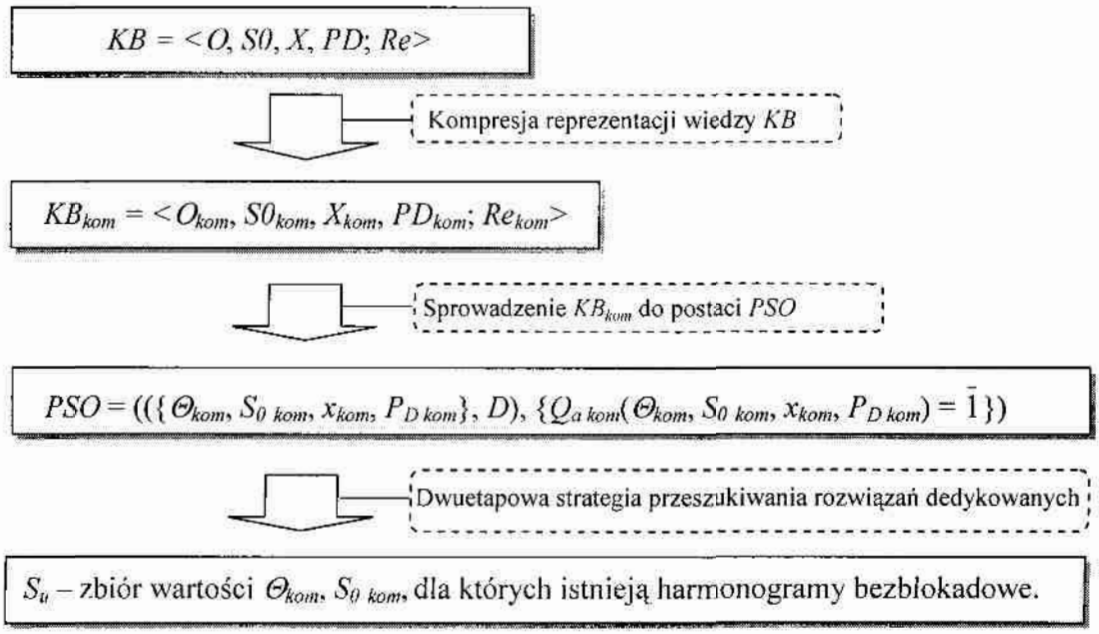
W rozdziale omówiono procedury wyznaczania warunków drugiego typu. Założono, że wykorzystywany schemat faktów umożliwia poprawne wnioskowanie.

Aby możliwe było poszukiwanie warunków $Fu(S_0,\Theta)$ wymagane jest by żądanie użytkownika zostało wyrażone w postaci zdania logicznego $Fy(x)$. Okazuje się że nie jest to zawsze możliwe. Rozważmy w pierwszej kolejności taki przypadek.

Właściwość „brak blokady i brak kolizji" nie daje się w sposób bezpośredni wyrazić jako zdanie logiczne $Fy(x)$ reprezentujące relacje między elementami sekwencji x. W rozdziale 4.2 wykazano, że jeśli ogólne założenia systemu transportowego zostaną wyrażone w postaci faktów $F_a(\Theta, S_0, x, P_D)$, implikujących spełnienie ograniczeń (4.2), (4.3), (4.4), (4.5), to w takim systemie, korzystając z Twierdzenia 4.1, można poszukiwać warunków gwarantujących brak blokady.

Na rysunku 4.31 przedstawiona została procedura wyznaczania warunków wystarczających oparta o wyznaczony schemat faktów, procedury kompresji, techniki programowania z ograniczeniami i dwuetapową strategię przeszukiwania.

W pierwszym etapie przedstawionej procedury, reprezentacja wiedzy $KB = <O, S0, X, PD; Re>$, opisująca ogólne zasady panujące w systemie (gdzie: relacja Re określona jest przez fakty $F_a(\Theta, S_0, x, P_D)$ wygenerowane ze schematu faktów), jest kompresowana przy użyciu metod zdefiniowanych w punkcie 4.3.2. W efekcie otrzymywana jest reprezentacja skompresowana $KB_{kom} = <O_{kom}, S0_{kom}, X_{kom}, PD_{kom}; Re_{kom}>$, w skład której wchodzą fakty: $F_{a\,kom}(\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom})$, gdzie: Θ_{kom}, $S_{0\,kom}$, x_{kom}, $P_{D\,kom}$ oznaczają zmienne Θ, S_0, x, P_D, wyrażone przy użyciu zasobów zastępczych R_{ZLi} i R_{ZWi}. Reprezentację taką zgodnie z własnościami opisanymi w punkcie 3.3, przedstawia się w postaci odpowiedniego $PSO = (\{\{\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom}\}, D), \{Q_{a\,kom}(\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom}) = \bar{1}\})$, gdzie: Θ_{kom}, $S_{0\,kom}$, x_{kom}, $P_{D\,kom}$ – zmienne decyzyjne, $D = \{D_{\Theta\,kom}, D_{0\,kom}, D_{x\,kom}, D_{PD\,kom}\}$ – zbiór dziedzin zmiennych decyzyjnych, $Q_{a\,kom}(\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom}) = \bar{1}$ – zbiór wartości logicznych faktów: $\{Q_{a1\,kom}(\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom}) = 1, \ldots, Q_{aK\,kom}(\Theta_{kom}, S_{0\,kom}, x_{kom}, P_{D\,kom}) = 1\}$.

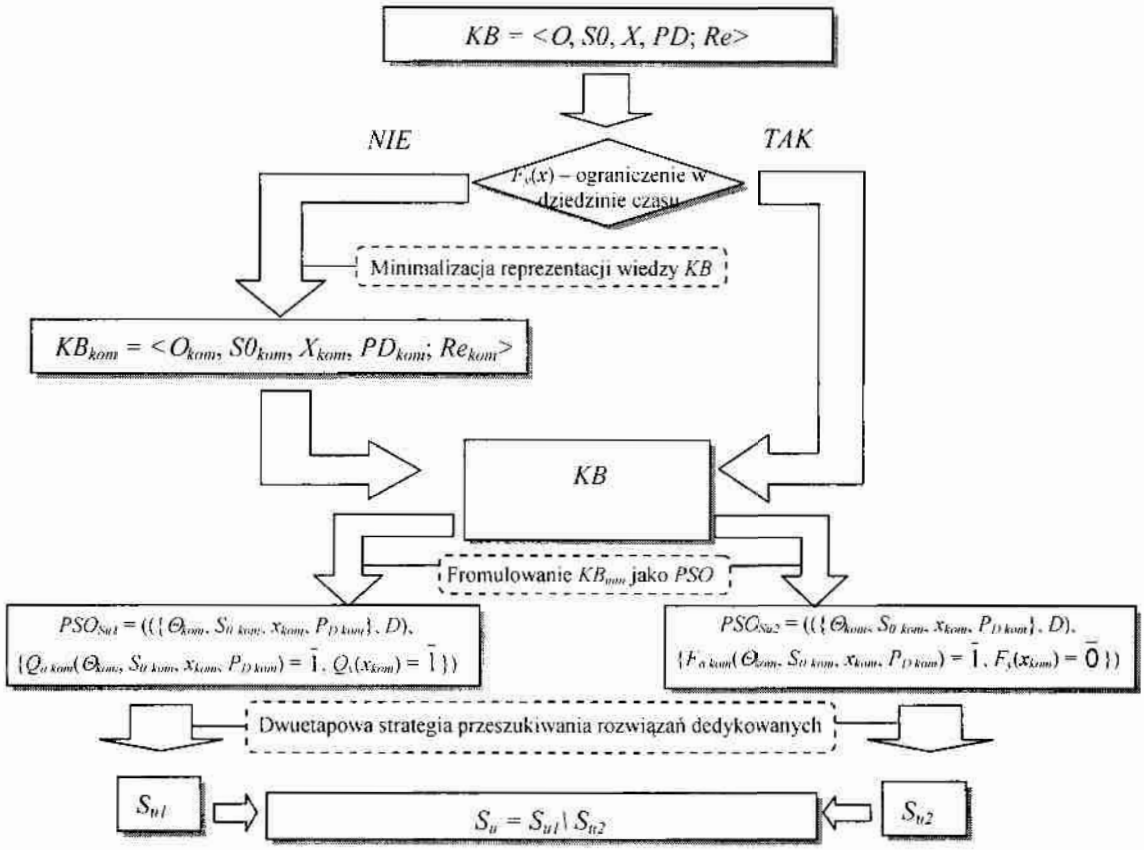


Rys. 4.31. Procedura wyznaczania warunków wystarczających gwarantujących brak blokady i kolizji

W tak sformułowanym problemie spełniania ograniczeń przyjęto, że ograniczenia reprezentują fakty $F_{a\,kom}$, którym przyporządkowuje się wartość logiczną 1. W otrzymanym *PSO* ograniczenia odpowiadają faktom F_a, a zatem podobnie jak fakty F_a gwarantują spełnienie ograniczeń (4.2), (4.3), (4.4), (4.5). Poszukiwanie warunków wystarczających polega na znalezieniu takich wartości zmiennych Θ_{kom}, $S_{0\,kom}$, dla których otrzymane harmonogramy są bezblokadowe. W tym celu wykorzystuje się opracowaną strategię dwuetapowego przeszukiwania rozwiązań dedykowanych oraz własności wynikające z Twierdzenia 4.1. Okazuje się, że w omawianym przypadku wystarczy tylko pierwszy etap opracowanej strategii. W etapie tym proces dystrybucji odnosi się tylko do zmiennych Θ_{kom}, S_{0kom}. W efekcie otrzymywane są rozwiązania dedykowane, tzn. rozwiązania o jednoelementowych dziedzinach zmiennych Θ_{kom}, S_{0kom}. W oparciu o rozmiar dziedzin $D_{x\,kom}$ zmiennych x, w otrzymanych rozwiązaniach dedykowanych, możliwa jest ocena czy dane rozwiązanie prowadzi do blokady. Jeżeli dla otrzymanego rozwiązania dedykowanego, które posiada określone wartości sekwencji Θ_{kom}, S_{0kom}, dziedziny wszystkich elementów sekwencji x_{kom} są jednoelementowe: $\forall D_{x\,\mathrm{kom}} \in D_{x\,\mathrm{kom}} : \|D_{xi\,\mathrm{kom}}\| = 1$, to wartości Θ_{kom}, S_{0kom} stanowią warunek gwarantujący, że harmonogram x_{kom} jest harmonogramem bezblokadowym.

W przypadku gdy dla określonego rozwiązania istnieje choć jeden element sekwencji x_{kom}, dla którego dziedzina jest wieloelementowa: $\exists D_{x\,\mathrm{kom}} \in D_{x\,\mathrm{kom}} : \|D_{xi\,\mathrm{kom}}\| > 1$ to wartościom sekwencji Θ_{kom}, S_{0kom}, odpowiadają wartości harmonogramu x_{kom} prowadzącego do blokady. Dziedziny wieloelementowe odpowiadają sytuacji, w której istnieje element sekwencji x_{kom} mogący przyjąć wartości z wieloelementowego zbioru. Wieloelementowe dziedziny elementów sekwencji x_{kom} świadczą o tożsamości równań stanu, co zgodnie z Twierdzeniem 4.1 oznacza stan blokady.

Na uwagę zasługuje fakt, że wyznaczonym wartościom Θ_{kom}, S_{0kom}, odpowiada jeden harmonogram x_{kom}, jednak ze względu na to, że jednemu zasobowi zastępczemu (R_{ZLi} lub R_{ZWi}) odpowiada zbiór zasobów, zatem jednej parze warunków Θ_{kom}, S_{0kom}, odpowiada zbiór harmonogramów x. W ten sposób wyznaczone warunki Θ_{kom}, S_{0kom}, mogą prowadzić do różnych harmonogramów x.



Rys. 4.32. Procedura wyznaczania warunków wystarczających gwarantujących spełnienie właściwości $Fy(x)$

Przedstawione postępowanie wykorzystywane jest w przypadku gdy użytkownik poszukuje warunków wystarczających gwarantujących brak występowania kolizji i blokad w systemach transportowych. W ogólności, poszukiwane mogą być warunki gwarantujące dowolną właściwość zdefiniowaną przez użytkownika. Procedurę wyznaczania warunków wystarczających przedstawiono na rysunku 4.32.

Pierwszym etapem jest określenie charakteru właściwości wyjściowej $Fy(x)$. Należy określić czy zadana właściwość opisuje relacje między zmiennymi x w dziedzinie czasu czy w dziedzinie stanów. Przez właściwość opisującą relacje w dziedzinie czasu rozumie się zdanie, które określa wartości zmiennych x, np. $x_1 > 4$.

Przez właściwość opisującą relację w dziedzinie stanów rozumie się zdanie, które określa kolejność elementów sekwencji x, np. $x_1 > x_2$. W zależności od charakteru własności $Fy(x)$, reprezentacja wiedzy może być kompresowana bądź też nie.

W dalszej kolejności reprezentację wiedzy przedstawia się w postaci dwóch problemów PSO_{Su1} oraz PSO_{Su2}. Do rozwiązania tych problemów wykorzystuje się strategię przeszukiwania rozwiązań dedykowanych. W efekcie otrzymuje się zbiory S_{u1} i S_{u2}, z których wyznaczany jest zbiór S_u będący zbiorem wartości Θ_{kom}, $S_{0\,kom}$ (lub Θ, S_0), dla których jest spełniona właściwość $Fy(x)$. Otrzymane warunki poza spełnieniem żądanej własności $Fy(x)$, gwarantują otrzymanie rozwiązań bezblokadowych i bezkolizyjnych. Okazuje się, że rozwiązania blokadowe są rozwiązaniami, które występują jednocześnie w zbiorze S_{u1} i S_{u2} (ze względu na charakter tożsamościowy, sekwencje x należą jednocześnie do obu zbiorów).

Zbiór $S_{u1} \backslash S_{u2}$ nie zawiera rozwiązań blokadowych (różnica zbiorów powoduje usunięcie elementów wspólnych).

W zależności od rodzaju własności $Fy(x)$ otrzymane warunki wystarczające, w postaci par Θ_{min}, S_{0min} (lub Θ, S_0), odpowiadają jednemu bądź wielu harmonogramom x.

Do tej pory rozważane były szczególne przypadki, w których w skład reprezentacji wiedzy KB wchodziły fakty $F_a(\Theta, S_0, x, P_D)$ opisujące ogólne zasady panujące w danym systemie transportowym. Każdy inny system transportowy (rozważanej klasy) może być także opisany przez indywidualne fakty dodatkowe $F_b(\Theta, S_0, x, P_D)$. Powstaje pytanie czy dodanie do reprezentacji wiedzy dodatkowych faktów F_b nie wpłynie na poprawność wyników procesu wnioskowania. W tym celu dokonywana jest weryfikacja faktów dodatkowych F_b. Weryfikacja polega na potraktowaniu faktów dodatkowych jako właściwość $Fy(x)$ i przeprowadzeniu procesu wyznaczania zbiorów S_{u1} i S_{u2}. W przypadku gdy otrzymane zbiory są rozłączne fakty F_b nie zakłócają procesu wnioskowania i mogą być swobodnie dodane do reprezentacji wiedzy. W przypadku gdy $S_{u2} \subset S_{u1}$ fakty można dodać do reprezentacji wiedzy ale pod warunkiem uzupełnienia jej jeszcze o dodatkowe ograniczenia wynikające z części wspólnej zbiorów S_{u1} i S_{u2}: $\Theta, S_0 \notin S_{u1} \cap S_{u2}$. Jeśli $S_{u1} \subseteq S_{u2}$ to fakty dodatkowe F_b są sprzeczne z faktami F_a lub prowadzą tylko do rozwiązań blokadowych.

W oparciu o przedstawione podejścia możliwe jest generowanie warunków wystarczających dla dowolnych właściwości $Fy(x)$ i dla dowolnych cech sytemu transportowego.

4.4. Podsumowanie

Proces weryfikacji bazy wiedzy obejmuje zagadnienia związane z formułowaniem schematu faktów tak by odpowiadały one ogólnej wiedzy na temat obiektu, oraz zagadnienia związane z wyznaczaniem warunków wystarczających (badanie spójności pod kątem zadanego pytania). W rozdziale przedstawiono realizację kolejnych etapów weryfikacji bazy wiedzy na przykładzie systemów transportowych.

W kontekście rozważanych systemów wykazano ograniczenia i twierdzenie umożliwiające budowę schematów faktów stanowiących opis wiedzy o systemie. Twierdzenie to jest konsekwencją zastosowania, do opisu systemów SWPC grafów żądań zasobowych stanowiących graficzną reprezentację (w dziedzinie czasu i zdarzeń) stanów systemu.

Zaproponowany schemat faktów jest jednym z wielu możliwych wariantów. Umożliwia on automatyczne wyznaczanie reprezentacji wiedzy KB w oparciu, o którą możliwe jest wyznaczanie warunków (stan początkowy i reguły priorytetowania) gwarantujących otrzymanie bezblokadowych i bezkolizyjnych harmonogramów pracy wózków samojezdnych. Poszukiwanie warunków wystarczających odbywa się poprzez rozwiązywanie problemu decyzyjnego.

W kontekście tego problemu opracowana została efektywna czasowo strategia wyznaczania zbiorów S_{u1}, S_{u2}, której istotą jest przeszukiwanie tylko części drzewa potencjalnych rozwiązań. Korzystanie z opracowanej strategii pozwala na zmniejszenie

wymaganej liczby obliczeń o kilka rzędów. Przedstawiono mechanizmy kompresji faktów i liczby zmiennych decyzyjnych poprzez minimalizację struktury połączeń systemu transportowego. Prezentowane metody kompresji mogą być wykorzystywane tylko dla warunków dotyczących pytań o charakterze stanowym.

Prezentowane w rozdziale rozwiązania stanowią niezbędne elementy do projektowania systemu interakcyjnego wspomagania decyzji. Wykorzystywane są one głównie w module **Reprezentacji wiedzy** (wyznaczenie reprezentacji wiedzy w oparciu o schemat faktów i parametry obiektu oraz ewentualną kompresję reprezentacji wiedzy), w module **Efektywnych strategii przeszukiwania** (implementacja dwuetapowej strategii przeszukiwania rozwiązań dedykowanych) oraz w module **Wyznaczania warunków gwarantujących istnienie rozwiązania** (wykorzystanie technik programowania z ograniczeniami do rozwiązania problemu decyzyjnego w oparciu o reprezentację wiedzy *KB* i efektywne czasowo strategie przeszukiwania). Działanie modułów odpowiada zaprezentowanym w rozdziale 4.3.3 procedurom wyznaczania warunków wystarczających (rysunki 4.31, 4.32).

Kolejne etapy weryfikacji bazy wiedzy zostały omówione w kontekście systemów transportowych, proponowana metodyka (obejmująca zagadnienia budowy schematów faktów, kompresji reprezentacji wiedzy, itp.) mimo to może być wykorzystywana w innych problemach.

5. Zadaniowo zorientowany interakcyjny system sterowania dyspozytorskiego

Celem przeprowadzanych badań jest opracowanie metodyki projektowania zadaniowo zorientowanych systemów interakcyjnego wspomagania podejmowania decyzji. Metodyka taka znajduje swoje użycie przy budowie systemu wspomagania sterowania dyspozytorskiego podsystemów transportowych ESP [18] , [22], [23].

Metodyka taka pozwala przyspieszyć budowę systemów podnosząc jednocześnie ich jakość (rozumianą jako możliwość formułowania wielu postaci pytań, możliwość formułowania dowolnego zakresu danych, itp.).

5.1. Metodyka projektowania komputerowych systemów wspomagania podejmowania decyzji

Przedstawione w rozdziałach 2, 3 i 4 badania, dotyczące zastosowania technik *CP* i metody logiczno-algebraicznej, stanowią podstawę budowy metodyki projektowania komputerowych systemów wspomagania podejmowania decyzji, obejmującej etapy:

- **Sformułowania problemu.** Wstępne sformułowanie zbioru pytań rutynowych oraz określenie zmiennych i relacji charakteryzujących funkcjonowanie obiektu.
- **Określenia postaci reprezentacji wiedzy *KB*.** Określenie zmiennych wejściowy u, pomocniczych w, wyjściowych y oraz relacji Re stanowiącej opis posiadanej wiedzy w postaci zbioru faktów. Dla przyjętej klasy problemu (reprezentowanej przez zbiór ogólnych zasad i reguł), formułowany jest tzw. schemat faktów $F_P(u,w,y,pr)$. Spośród różnych postaci schematów faktów poszukiwany jest ten, który gwarantuje istnienie odpowiedzi na sformułowany zbiór pytań.
- **Wyboru języka programowania *CP*.** Wybór takiego języka programowania *CP*, który umożliwia implementację reprezentacji *KB* przyjętego problemu.
- **Implementacji skryptu obliczeniowego.** Przygotowanie kodu programu, obejmującego wybraną reprezentację wiedzy *KB*, minimalizację schematu faktów, wyznaczenie strategii poszukiwania warunków wystarczających w wybranym języku *CP*.
- **Poszukiwania warunków wystarczających.** Poszukiwanie warunków, gwarantujących istnienie odpowiedzi na zadane pytania.
- **Wyboru strategii rozwiązywania *PSO*.** Wybór efektywnej czasowo strategii poszukiwania rozwiązań dopuszczalnych.
- **Budowy interfejsu.** Budowa nakładki umożliwiającej wprowadzanie i wyprowadzanie danych do skryptu obliczeniowego, w wybranym pakiecie programowania.

Zaproponowane podejście koncentruje się na specyfikacji zbioru ograniczeń, reprezentujących wiedzę o problemie, wykorzystując w tym celu ograniczenia wynikające z doświadczenia programisty oraz dostępne pakiety oprogramowania *CP*.

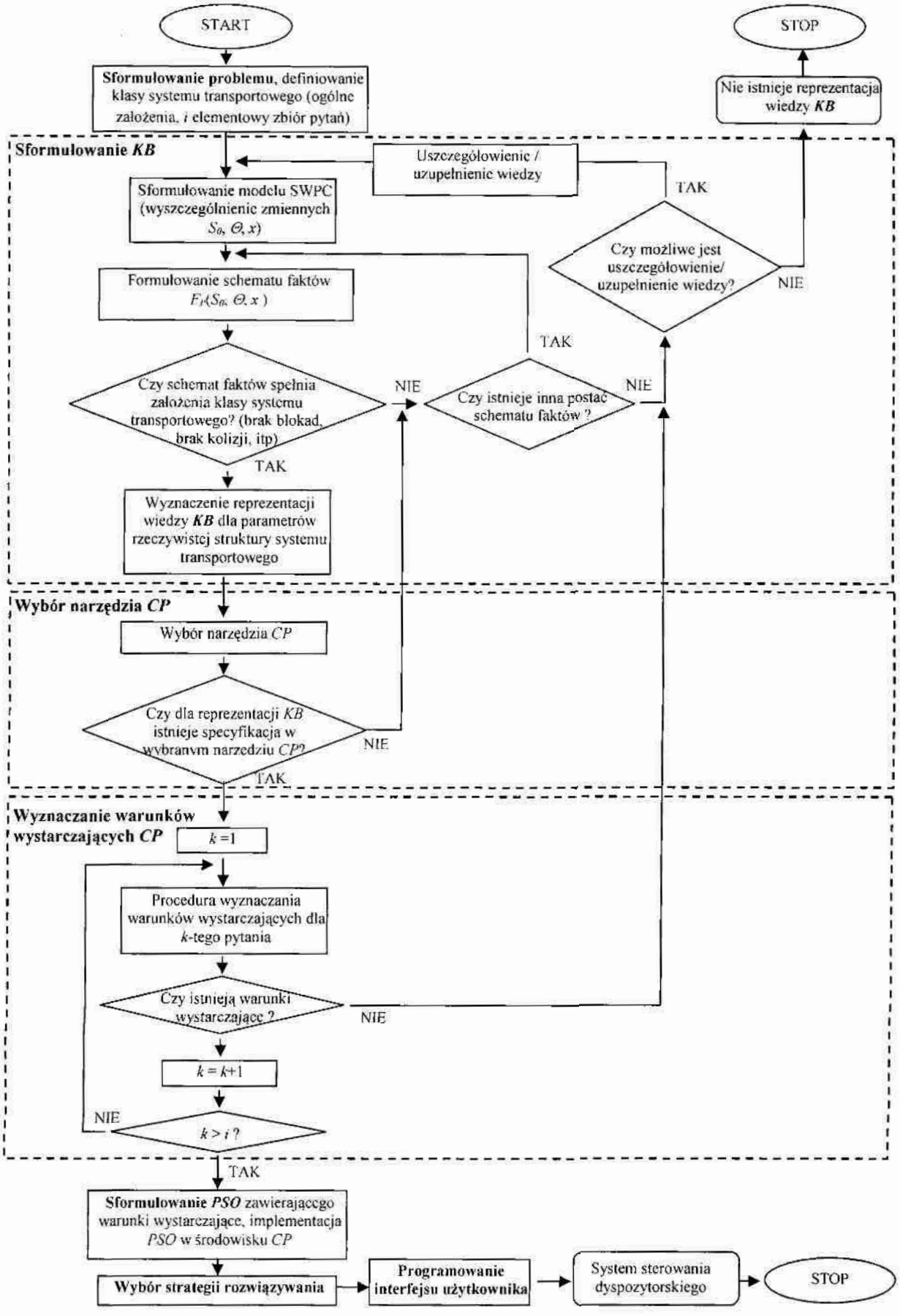
Sformułowane zagadnienia prowadzą do procedury projektowania zdaniowo zorientowanych systemów interakcyjnego wspomagania decyzji. Procedura ilustrująca proces projektowania systemu sterowania dyspozytorskiego przedstawiona została na rysunku 5.1. Na uwagę zasługują etapy specyfikowania reprezentacji wiedzy (*KB*), wyboru języka programowania z ograniczeniami (*CP*) oraz poszukiwania warunków wystarczających.

Formułowanie reprezentacji wiedzy jest procedurą iteracyjnie powtarzaną do momentu uzyskania takiej postaci schematu faktów, która gwarantuje spełnienie wszystkich założeń (systemu transportowego) i ogólnych zasad w kontekście parametrów S_0, Θ, x. Dla systemów transportowych należących do klasy zdefiniowanej w rozdziale 4, poszukiwana jest taka postać schematu faktów, dla której spełnione są ograniczenia (4.1), (4.2), (4.3), (4.4), a tym samym spełnione jest Twierdzenie 4.1. W przypadku, gdy przyjęta postać schematu faktów jest niewłaściwa, wówczas podejmowana jest próba przeformułowania schematu faktów. Dopiero w momencie, gdy operacja taka staje się niemożliwa wymagane jest uszczegółowienie lub uzupełnienia wiedzy dotyczącej problemu. Jeżeli dla zadanej wiedzy nie istnieje schemat faktów, który spełnia powyższe ograniczenia to budowa systemu wspomagania decyzji przy użyciu omawianych podejść i narzędzi nie jest możliwa.

Analogicznie do etapu, w którym poszukiwana jest postać reprezentacji wiedzy, poszukiwane jest środowisko programowania z ograniczeniami. Wymagane jest by zbiór faktów wpływających na postać relacji *Re* wchodzącej w skład *KB* mógłby być wyrażony w postaci odpowiedniego zbioru ograniczeń *C*. W przypadku gdy nie istnieje środowisko *CP* umożliwiające implementację zadanej reprezentacji wiedzy, należy wówczas dla rozważanego problemu sformułować nową postać reprezentacji wiedzy.

Etap wyznaczania warunków wystarczających polega na poszukiwaniu warunków wystarczających dla każdego pytania ze zbioru pytań rutynowych. Dla systemów transportowych należących do klasy zdefiniowanej w rozdziale 4, poszukuje się warunków zgodnie z procedurami przedstawionymi na rysunkach 4.31 i 4.32. Jeżeli istnieje co najmniej jedno pytanie, dla którego niemożliwe jest wyznaczenie warunków wystarczających, to należy poszukiwać innej postaci reprezentacji wiedzy (*KB*).

Proponowana metodyka obejmuje etapy specyfikacji reprezentacji wiedzy, dobór narzędzi *CP* oraz strategii przeszukiwania. Umożliwia wybór takiej specyfikacji problemu, która gwarantuje spełnienie ograniczeń wynikających z przyjętych zasad i ograniczeń rozważanych systemów. Należy podkreślić, że istotnym aspektem projektowania systemu jest potrzeba rozdzielenia modułów obliczeniowych i modułów interfejsu. Rozdzielenie modułów umożliwia wymianę języka jądra obliczeń na inny język *CP*, przy zachowaniu tego samego interfejsu użytkownika. Z drugiej strony, moduł obliczeniowy może stanowić jeden z wielu elementów większego systemu wspomagania decyzji (np. systemu sterowania operacyjnego).



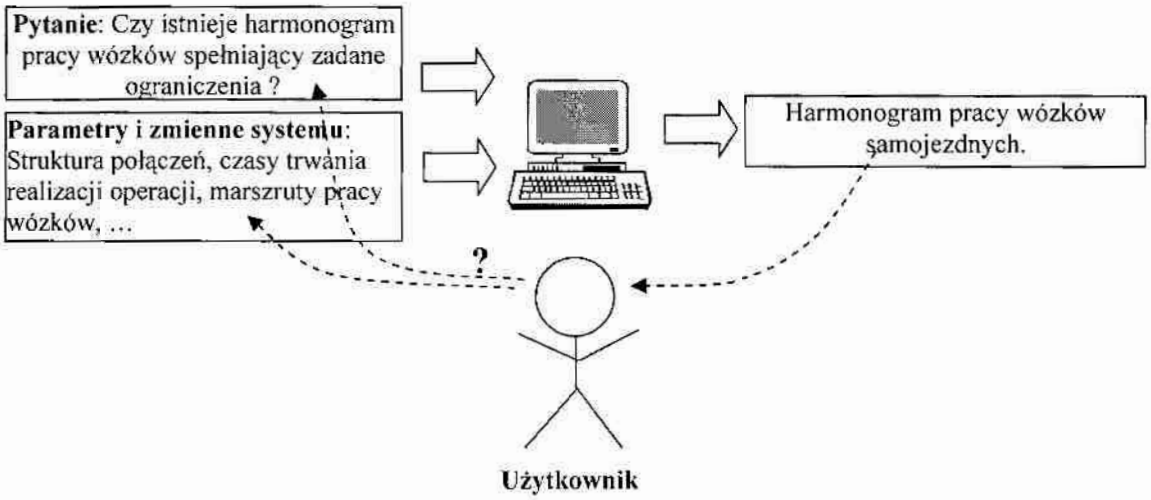
Rys. 5.1. Procedura projektowania systemu sterowania dyspozytorskiego

5.2. Budowa systemu

Procedura przedstawiona na rysunku 5.1 wykorzystana została przy budowie *Systemu Sterowania Dyspozytorskiego* (*SSD*). *System Sterowania Dyspozytorskiego* w systemie transportowym wykorzystuje procedury wyznaczania warunków wystarczających, strategie przeszukiwania przestrzeni potencjalnych rozwiązań oraz model systemu interakcyjnego wspomagania decyzji. Zadaniem opracowanego systemu jest udzielanie odpowiedzi na pytania decydenta, w szczególności te dotyczące bilansowania jego potrzeb z ograniczeniami systemu transportowego.

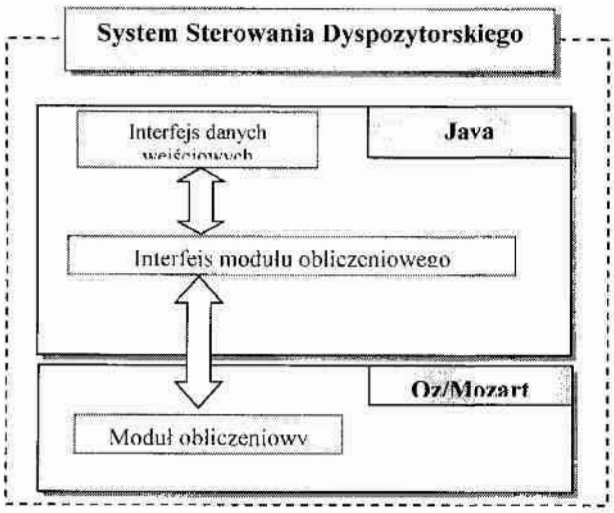
Zbudowany *System Sterowania Dyspozytorskiego* wspomaga decyzje dyspozytora podsystemów transportowych będących częścią elastycznych systemów produkcyjnych spełniających założenia zdefiniowane w rozdziale 4. System ten pozwala na wyznaczenie, w trybie interakcyjnym, dopuszczalnego harmonogramu pracy wózków samojezdnych, spełniającego ograniczenia zadane przez użytkownika systemu. Wyznaczenie harmonogramu polega na znalezieniu planu, obejmującego terminy rozpoczęcia i ukończenia operacji elementarnych, stan początkowy procesów i reguły obsługi procesów na zasobach współdzielonych, itp.

Wyznaczony plan spełnia ograniczenia wynikające z infrastruktury transportowej oraz gwarantuje bezkolizyjną i bezblokadową pracę systemu. Ideę działania systemu zilustrowano na rysunku 5.2.



Rys. 5.2. Idea działania *Systemu Sterowania Dyspozytorskiego*

Przyjęta struktura systemu składa się z dwóch części (rysunek 5.3). Pierwszą stanowi **moduł obliczeniowy** wykorzystujący techniki *CP*, drugą zaś **interfejs użytkownika**. Interfejs ten pozwala na szybkie i łatwe wprowadzanie danych oraz wizualizację wyników działania modułu obliczeniowego.



Rys. 5.3. Struktura przepływu informacji w *Systemie Sterowania Dyspozytorskiego*

SSD bazuje na pakiecie **Oz Mozart** oraz języku Java. W systemie tym można wyróżnić 2 warstwy, zróżnicowane pod względem realizowanych zadań oraz zakresu ingerencji użytkownika. Pierwszą z nich stanowi **interfejs użytkownika** (który jest dzielony na: **interfejs danych wejściowych** i **interfejs modułu obliczeniowego**). Rozwiązanie takie pozwala modyfikować wartości zmiennych decyzyjnych oraz parametry **modułu obliczeniowego**. **Moduł obliczeniowy** stanowi warstwę drugą. Stanowi on rodzaj biblioteki zadaniowo zorientowanych procedur wykorzystywanych podczas wyznaczania warunków wystarczających jak i odpowiedzi na zadane pytania. Użytkownik ma ograniczony wpływ na działanie tych procedur, tzn. może wybrać parametry oczekiwanego wyniku (poprzez definiowanie własnych ograniczeń) oraz wpływać na liczbę wyznaczanych warunków wystarczających.

Warstwa **modułu obliczeniowego** została zrealizowana w języku **Oz Mozart**, pozostałe przy użyciu języka programowania Java. Rozwiązanie to, z jednej strony, pozwala przygotować program przyjazny dla użytkownika, z drugiej zaś, stosować techniki *CP* do poszukiwania rozwiązania problemu. Działanie **modułu obliczeniowego** realizowane jest w tle, a użytkownik (decydent) systemu może modyfikować arbitralnie zadany zbiór parametrów procesu obliczeń.

Główny element programu stanowi **interfejs użytkownika**. Pozwala on w prosty i intuicyjny sposób wprowadzać dane do obliczeń. Wyróżniono w nim składniki pozwalające na akwizycję danych dotyczących parametrów infrastruktury transportowej, stanu początkowego, reguł obsługi ruchu, parametrów środków transportu oraz określania parametrów **modułu obliczeniowego**.

W części dotyczącej infrastruktury transportowej podsystemu użytkownik wprowadza do *SSD* strukturę istniejących tras transportowych wraz z czasami realizacji poszczególnych operacji na określonych zasobach.

W części dotyczącej stanu początkowego użytkownik określa stany początkowe poszczególnych procesów oznaczające lokalizację początkową poszczególnych wózków

samojezdnych. Może również zadawać reguły obsługi ruchu na określonych zasobach współdzielonych.

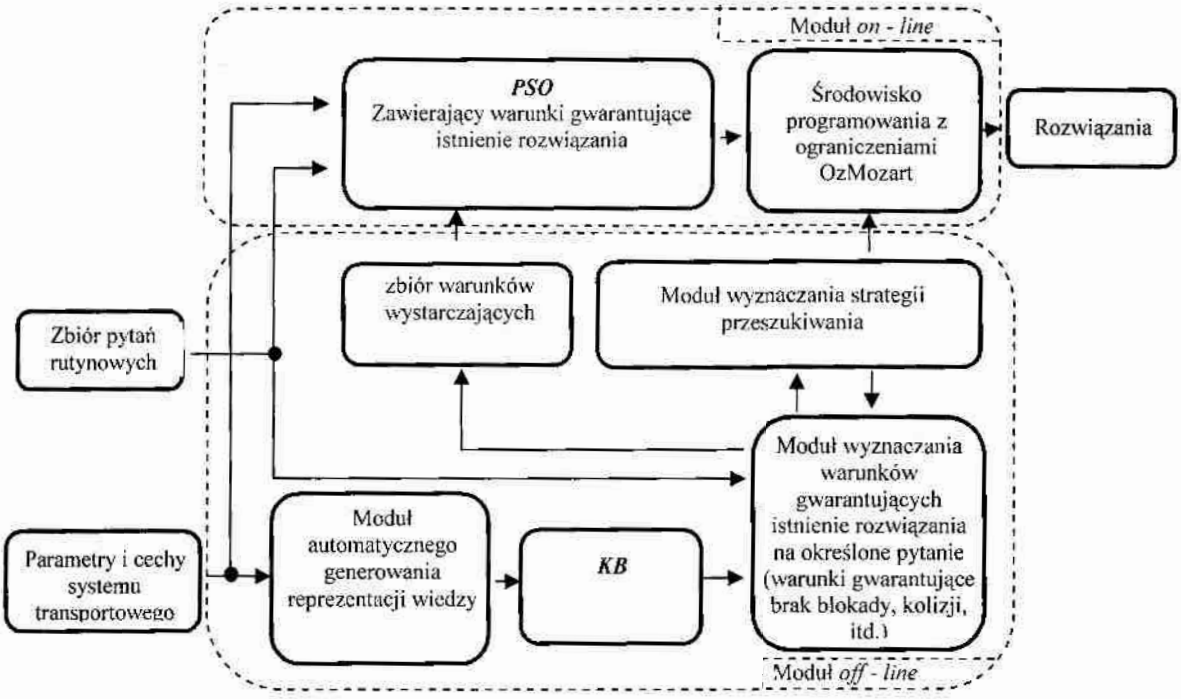
W części dotyczącej parametrów środków transportu decydent wprowadza marszruty poszczególnych wózków samojezdnych oraz liczbę cykli realizowanych przez dany wózek w zadanym oknie czasowym.

W części dotyczącej parametrów **modułu obliczeniowego (interfejs modułu obliczeniowego)** wprowadzane są ograniczenia dotyczące sytemu transportowego związane z zadanym pytaniem, określany jest czas trwania okna czasowego oraz parametry dla procedur liczących i procedur prezentujących wyniki obliczeń (diagramy Gantt'a).

Wyznaczenie harmonogramu pracy wózków samojezdnych wymaga:

- wprowadzenia środków transportu,
- wprowadzenia parametrów infrastruktury transportowej,
- wprowadzenia stanów początkowych sieci i reguł obsługi ruchu,
- zdefiniowania rozmiaru okna czasowego oraz dodatkowych ograniczeń,
- wprowadzenia ograniczeń użytkownika dotyczących na przykład wartości horyzontu realizacji operacji, ograniczeń kolejności realizacji operacji itp.,
- uruchomienia obliczeń związanych z wyznaczeniem warunków wystarczających
- uruchomienia obliczeń związanych z poszukiwanie rozwiązania dopuszczalnego
- analizy otrzymanych wyników.

Poszukiwanie rozwiązania dopuszczalnego realizowane jest w oparciu o wyznaczone wcześniej warunki wystarczające. Użytkownik poprzez definiowanie własnych ograniczeń ma możliwość analizy różnych wariantów pracy systemu transportowego. Wybór różnych wariantów poszukiwania rozwiązania zapewnia przyjęta struktura interakcyjnego *SSD* (rysunek 5.4).



Rys. 5.4. Struktura *Systemu Sterowania Dyspozytorskiego*

Umożliwia ona, w zależności od postaci pytań i cech systemu transportowego, formułowanie problemu w postaci *PSO* (zawierającego warunki wystarczające) oraz wykorzystanie do poszukiwania rozwiązań efektywnych strategii przeszukiwania (opartych na technikach *CP*). Struktura ta stanowi uszczegółowienie struktury interakcyjnego systemu wspomagania decyzji z rysunku 2.4.

Struktura systemu została podzielona na dwa moduły. Pierwszy moduł, pracujący zwykle w trybie *off-line*, odpowiedzialny jest za proces wyznaczania warunków wystarczających gwarantujących istnienie odpowiedzi dla zadanego zbioru pytań i parametrów systemu transportowego. Wyznaczone warunki umieszczane są jako dodatkowe ograniczenia w problemie *PSO*. Generowanie postaci reprezentacji wiedzy *RW* w oparciu o posiadane parametry odbywa się zgodnie z procedurą przedstawioną na rysunku 5.1. Proces wyznaczania warunków wystarczających (rozwiązanie problemu decyzyjnego) odbywa się według procedury przedstawionej na rysunku 4.32.

Drugi moduł odpowiedzialny jest za wyznaczenie odpowiedzi na zadane pytanie poprzez rozwiązanie problemu *PSO*. Gwarantuje odpowiedź na zadane pytania w trybie *on-line*.

Oba moduły stanowią integralną całość, mimo to mogą stanowić one również niezależnie pracujące programy. Moduł *off-line* może być wykorzystywany do budowy systemów interakcyjnego wspomagania decyzji, z kolei moduł *on-line* może być traktowany jako aplikacja stanowiąca zorientowany zadaniowo system interakcyjnego wspomagania decyzji wyposażona w efektywne strategie przeszukiwania.

5.3. Działanie systemu

Działanie *Systemu Sterowania Dyspozytorskiego* zilustrowano na wybranych przykładach problemów harmonogramowania pracy wózków w systemach transportowych.

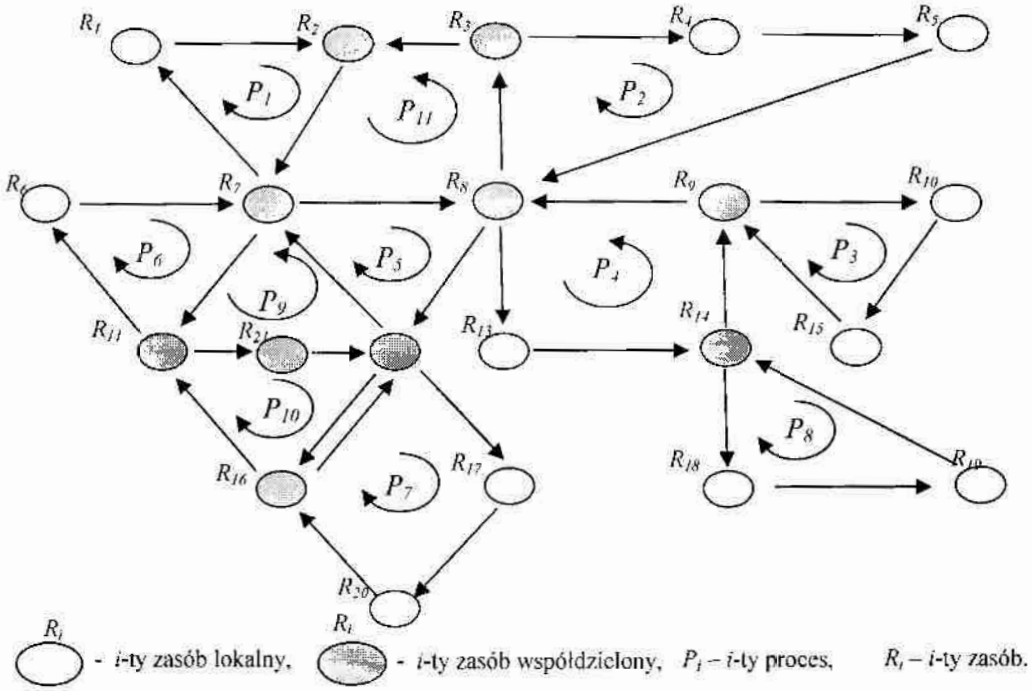
Problem 1.

Dany jest system wózków samojezdnych, którego strukturę ilustruje rysunek 5.5. Procesy $P_1, P_2, \ldots, P_{11}$, odpowiadają trasom 11 wózków transportowych $Tw_1, Tw_2, \ldots, Tw_{11}$. Wózki obsługiwane są przez punkty obsługi $R_1, R_2, \ldots, R_{21}$ - zasoby systemu. Swoje operacje wózki realizują cyklicznie.

Znane są marszruty poszczególnych procesów oraz czasy jednostkowe operacji elementarnych:

$P_1 = (R_1, R_2, R_7)$, $T_1 = (1,2,3)$,
$P_2 = (R_3, R_4, R_5, R_8)$, $T_2 = (1,2,3,1)$,
$P_3 = (R_9, R_{10}, R_{15})$, $T_3 = (2,1,3)$,
$P_4 = (R_8, R_{13}, R_{14}, R_9)$, $T_4 = (1,2,3,1)$,
$P_5 = (R_7, R_8, R_{12})$, $T_5 = (1,2,3)$,
$P_6 = (R_6, R_7, R_{11})$, $T_6 = (1,2,3)$,
$P_7 = (R_{12}, R_{17}, R_{20}, R_{16})$, $T_7 = (1,2,3,1)$,
$P_8 = (R_{14}, R_{18}, R_{19})$, $T_8 = (1,2,3)$,

$P_9 = (R_7, R_{11}, R_{21}, R_{12})$, $T_9 = (1,2,3,1)$,
$P_{10} = (R_{11}, R_{21}, R_{12}, R_{16})$, $T_{10} = (1,2,3,1)$,
$P_{11} = (R_2, R_7, R_8, R_3)$, $T_{11} = (1,2,3,1)$.

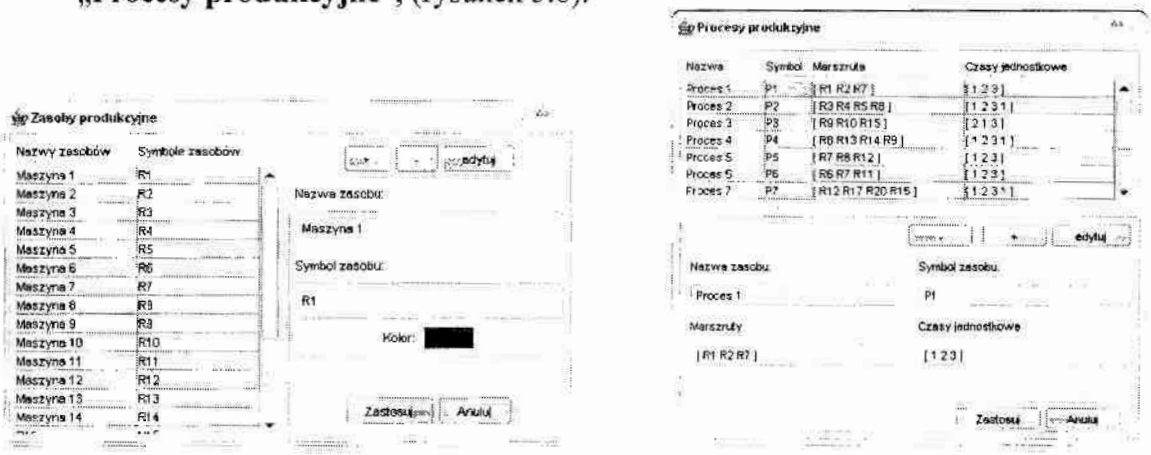


Rys. 5.5. Struktura SWPC systemu transportowego

Użytkownik systemu poszukuje odpowiedzi na pytanie: Czy możliwa jest bezblokadową i bezkolizyjna praca wózków?

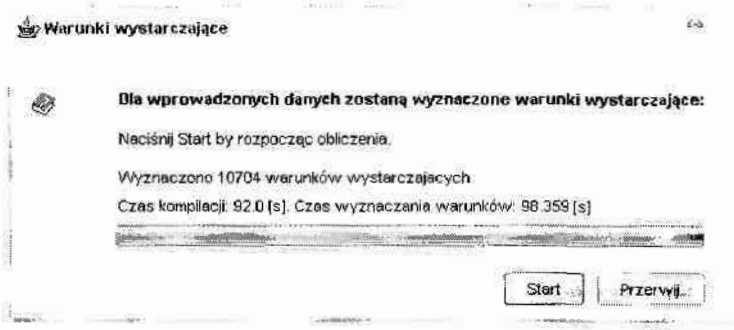
Odpowiedź na tak postawione pytanie wymaga:

- Wprowadzenia danych do systemu: wykorzystując okna: „**Zasoby produkcyjne**", „**Procesy produkcyjne**", (rysunek 5.6).

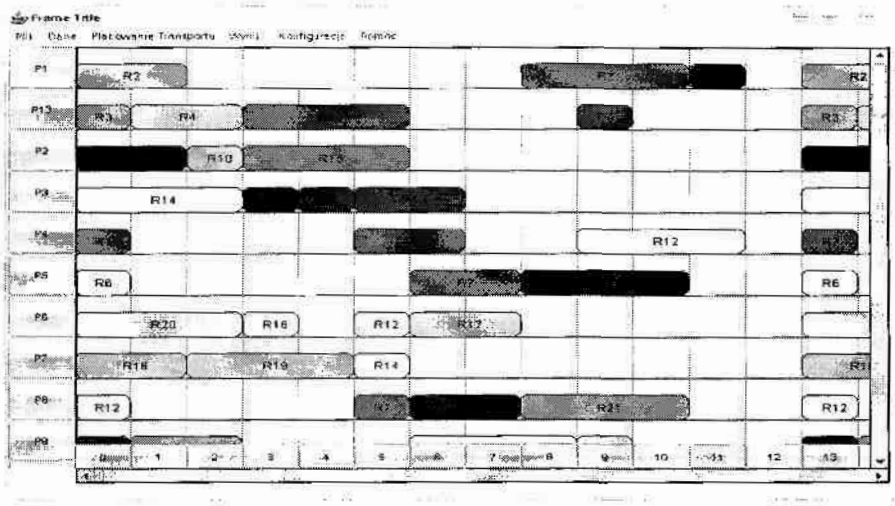


Rys. 5.6. Okna do wprowadzania danych: a) okno zasobów produkcyjnych b) okno procesów produkcyjnych

- Wyznaczania warunków wystarczających: po wprowadzeniu danych wyznaczane są warunki wystarczające, okno: „**Warunki wystarczające**" (rysunek 5.7).



Rys. 5.7. Okno do wyznaczania warunków wystarczających



Legenda:

R12 - graficzna reprezentacja czasu pracy procesu na zasobie R_{12},

Pi - proces P_i.

Rys. 5.8. Przykładowy harmonogram pracy wózków

Wyznaczonych zostało w czasie 200 sekund 10704 alternatywnych warunków wystarczających (stany początkowe i reguły priorytetowania).

Dla wyznaczenia przykładowego harmonogramu pracy, wykorzystywane jest okno: „**Wyznacz harmonogram**". Wyznaczenie przykładowego harmonogramu (rysunku 5.8) wymagało 4 sekund.

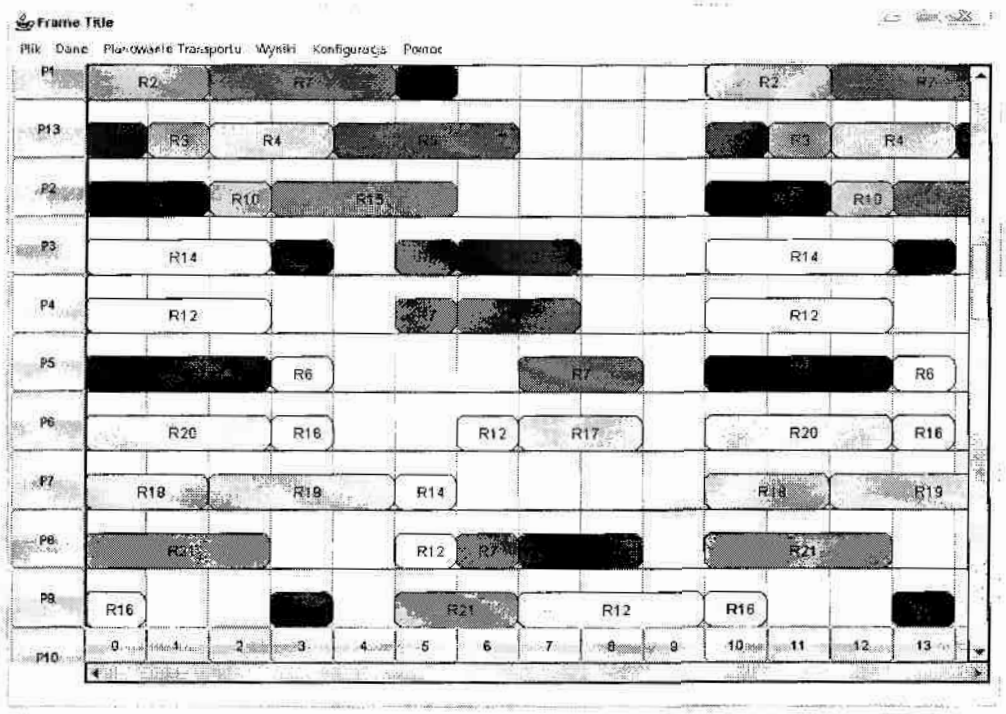
Problem 2.

Dany jest system transportowy zdefiniowany w **Problemie 1.** Użytkownik poszukuje harmonogramu, w którym wszystkie operacje elementarne będą realizowane w cyklach nie większych niż 10 jednostek czasu. Należy odpowiedzieć na pytanie: Czy istnieje harmonogram pracy wózków w cyklach nie przekraczających 10 jednostek czasu ?

W tym celu wykorzystywane są warunki wystarczające już poprzednio wyznaczone. Dla wprowadzenia ograniczenia mówiącego, że cykl nie może przekraczać 10 jednostek czasu wykorzystywane jest okno: „**Ograniczenia własne**".

Rys. 5.9. Okno Ograniczenia własne

Następnie, korzystając z okna „**Wyznacz harmonogram**" uruchomiana jest procedura wyznaczania harmonogramu. Harmonogram spełniający zadane ograniczenie (rysunek 5.10) uzyskiwany jest w czasie 4 sekund.



Legenda:

R12 - graficzna reprezentacja czasu pracy procesu na zasobie R_{12}.

Pi - proces P_i.

Rys. 5.10. Harmonogram z cyklem nieprzekraczającym 10 jednostek czasu

Problem 3.

Dany jest system transportowy jak w **Problemie 1.** Użytkownik poszukuje harmonogramu realizacji poszczególnych operacji, w którym spełnione będą następujące ograniczenia:

- Proces P_1 może rozpocząć pracę na zasobie R_7 pod warunkiem, że operacja procesu P_5 zrealizowana została już na zasobie R_7.
- Operacja procesu P_7 na zasobie R_{14}, musi rozpocząć się przed operacją procesu P_3 na zasobie R_8.

Wykorzystywane są warunki wystarczające już poprzednio wyznaczone. Dla wprowadzenia do systemu zadanych ograniczeń wykorzystywane jest okno **„Ograniczenia własne"** (zgodnie z rysunkiem 5.11).

Rys. 5.11. Okno Ograniczenia własne

Ograniczenia mają postać:

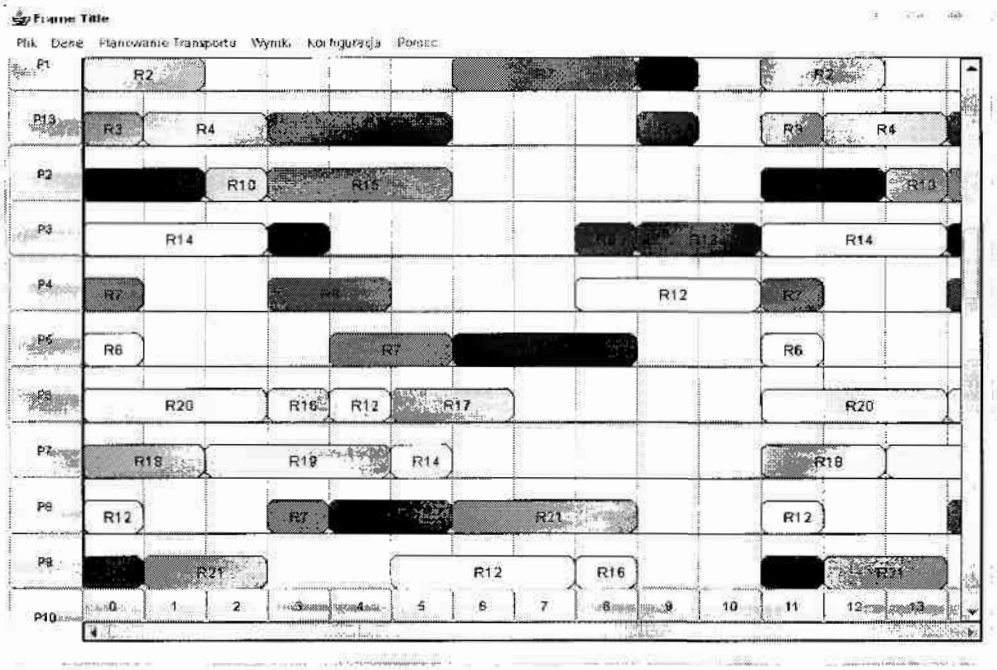
P1.R7 >: P5.R7,

P7.R14 <: P3.R8.

gdzie: Pi.Rj – oznacza termin rozpoczęcia operacji procesu P_i na zasobie R_j.

Następnie, wykorzystując okno: **„Wyznacz harmonogram"** uruchamiana jest procedura wyznaczania harmonogramu. Harmonogram spełniający zadane ograniczenia (rysunek 5.12) uzyskiwany jest w czasie 4 sekund. Wykorzystując okno **„Ograniczenia własne"** użytkownik ma możliwość wprowadzania własnych ograniczeń, tak na etapie wyznaczania warunków, jak i na etapie wyznaczania gotowych harmonogramów (postacie ograniczeń jakie mogą być zadawane przez użytkownika zostały opisane w załączniku A).

Przedstawione problemy ilustrują niektóre możliwości zrealizowanego *Systemu Sterowania Dyspozytorskiego*. Na uwagę zasługuje fakt, że w przypadku posiadania warunków wystarczających odpowiedź systemu jest wyznaczana w czasie poniżej 5 sekund. Możliwa jest zatem współpraca systemu z użytkownikiem w trybie interakcyjnym.



Legenda:

R12 - graficzna reprezentacja czasu pracy procesu na zasobie R_{12},

Pi - proces P_i.

Rys. 5.12. Harmonogram spełniający ograniczenia: P1.R7 >: P5.R7; P7.R14 <: P3.R8

5.4. Scenariusze typowych problemów

5.4.1. Pytania rutynowe

W kontekście rozważanej klasy systemów transportowych opracowany *SSD* pozwala na analizę, m.in. następujących wariantów problemów wspomagania decyzji:

a) Dany jest system transportowy o określonych parametrach. Dana jest liczba środków transportu o znanych parametrach. Znane są ograniczenia czasowe i kolejnościowe wynikające z indywidualnych potrzeb użytkownika, np. ograniczenia technologiczne, ograniczenia określające kolejność realizacji poszczególnych operacji, ograniczenia dotyczące czasu trwania cyklu, itp. Poszukiwana jest odpowiedź na pytanie:

Czy praca wózków może odbywać się bezkolizyjnie i bezblokadowo? Jeśli tak, to jaką postać ma dopuszczalny harmonogram pracy wózków?

W celu odpowiedzi, *SSD* obsługiwany jest według następującej procedury:

1. Otworzyć okno „**Zasoby produkcyjne**" (Menu → Dane → Zasoby), za pomocą okna wprowadzić dane dotyczące zasobów (rysunek 5.6).
2. Otworzyć okno „**Procesy produkcyjne**" (Menu → Dane → Procesy) za pomocą okna wprowadzić dane dotyczące procesów, marszruty produkcyjne, czasy trwania elementarnych operacji (rysunek 5.6).
3. Otworzyć okno „**Ograniczenia własne**" (Menu → Dane → Ograniczenia własne) w zakładce „**Ograniczenia dla warunków**" określić (jeśli to

konieczne) ograniczenia czasu trwania okna, oraz wprowadzić ograniczenia czasowe i kolejnościowe użytkownika (rysunek 5.11).

4. Otworzyć okno „**Warunki wystarczające**" (Menu → Planowanie Transportu → Wyznacz warunki), naciskając przycisk „Start" uruchomić proces wyznaczenia warunków wystarczających. Po zakończeniu procesu obliczeń zamknąć okno. W ten sposób użytkownik wyznacza warunki wystarczające gwarantujące w zadanym systemie pracę wózków bez kolizji i blokad.
5. Otworzyć okno „**Ograniczenia własne**" (Menu → Dane → Ograniczenia własne) w zakładce „ograniczenia dla harmonogramów" określić ograniczenia czasu trwania okna, oraz wprowadzić ograniczenia czasowe użytkownika (rysunek 5.11).
6. Otworzyć okno „**Harmonogram**" (Menu → Planowanie transportu → Wyznacz harmonogram), naciskając przycisk Start uruchomić proces wyznaczenia warunków wystarczających. Po zakończeniu procesu obliczeń zamknąć okno. W ten sposób program wyznacza harmonogram spełniający warunki wystarczające i ograniczenia użytkownika.
7. W celu ilustracji wyniku wybrać Menu → Wyniki → Harmonogram dopuszczalny (rysunek 5.8), w efekcie pojawi się harmonogram stanowiący jednocześnie odpowiedź na zadane pytanie.

b) Dany jest system transportowy o określonych parametrach. Dana jest liczba środków transportu o znanych parametrach. Znane są warunki wystarczające gwarantujące bezblokadową i bezkolizyjną pracę systemu, a także ograniczenia zadane przez użytkownika (przyjmuje się że, zrealizowane zostały już punkty a.1-a.3).

- **Czy zadana postać reguł obsługi wózków na określonych zasobach współdzielonych oraz dodatkowo wprowadzone ograniczenia, gwarantują bezkolizyjną i bezblokadową pracą systemu?**

W celu odpowiedzi, *SSD* obsługiwany jest według następującej procedury:

1. Otworzyć okno „**Wyznaczone warunki**" (Menu → Wyniki → Warunki wystarczające), za pomocą przycisków „+", „-", „edycja", uzupełnić zbiór reguł o własne reguły obsługi ruchu (rysunek B.15).
2. Otworzyć okno „**Ograniczenia własne**" (Menu → Dane → Ograniczenia własne) w zakładce „**Ograniczenia dla harmonogramów**" określić ograniczenia czasu trwania okna, oraz wprowadzić ograniczenia czasowe użytkownika (rysunek 5.11).
3. Otworzyć okno „**Harmonogram**" (Menu → Planowanie transportu → Wyznacz harmonogram), naciskając przycisk Start uruchomić proces wyznaczenia warunków wystarczających. Po zakończeniu procesu obliczeń zamknąć okno. W ten sposób program wyznaczy harmonogram spełniający warunki wystarczające i ograniczenia użytkownika.
4. W celu ilustracji wyniku wybrać Menu → Wyniki → Harmonogram dopuszczalny (rysunek 5.8), w efekcie pojawi się harmonogram stanowiący

jednocześnie odpowiedź na zadane pytanie (jeśli oczywiście istnieje harmonogram spełniający zadane ograniczenia).

- **Czy zmiana czasów realizacji operacji elementarnych gwarantuje spełnienie zdanych ograniczeń (brak blokady i kolizji, arbitralnie zadanych ograniczeń użytkownika)?**

W celu odpowiedzi, *SSD* obsługiwany jest według następującej procedury:

1. Otworzyć okno „**Zasoby produkcyjne**" (Menu → Dane → Procesy), za pomocą okna zmodyfikować dane dotyczące czasów operacji elementarnych (rysunek 5.6).
2. Otworzyć okno „**Harmonogram**" (Menu → Planowanie transportu → Wyznacz harmonogram), naciskając przycisk „Start" uruchomić proces wyznaczenia warunków wystarczających. Po zakończeniu procesu obliczeń zamknąć okno. W ten sposób program wyznacza harmonogram spełniający warunki wystarczające i ograniczenia użytkownika.
3. W celu ilustracji wyniku wybrać Menu → Wyniki → Harmonogram dopuszczalny (rysunek 5.8), w efekcie pojawi się harmonogram stanowiący jednocześnie odpowiedź na zadane pytanie (jeśli oczywiście istnieje harmonogram spełniający zadane ograniczenia).

c) Dany jest system transportowy o określonych parametrach. Dana jest liczba środków transportu o znanych parametrach. Znane są warunki wystarczające gwarantujące spełnienie zadanych ograniczeń.

- **Czy zmiana struktury transportowej (usunięcie/dodanie zasobu), gwarantuje spełnienie ograniczeń?**
- **Czy dodanie/usunięcie wózka samojezdnego wraz z określoną marszrutą realizacji operacji, nie spowoduje kolizji, blokady w systemie?**
- **Czy zmiana marszruty określonego wózka nie spowoduje kolizji, blokady w systemie?**

W celu odpowiedzi na pytania, *SSD* obsługiwany jest według następującej procedury:

1. Otworzyć okno „**Opcje**" (Menu → Konfiguracja → Opcje), w ustawieniach okna zmienić wartość maksymalnej liczby warunków na 1 (rysunek B.1).
2. Otworzyć okno „**Zasoby produkcyjne**" (Menu → Dane → Zasoby), za pomocą okna wprowadzić dane dotyczące zasobów (rysunek 5.6).
3. Otworzyć okno „**Procesy produkcyjne**" (Menu → Dane → Procesy) za pomocą okna wprowadzić dane dotyczące procesów, marszruty produkcyjne, czasy trwania elementarnych operacji (rysunek 5.6).
4. Otworzyć okno „**Ograniczenia własne**" (Menu → Dane → Ograniczenia własne) w zakładce „**Ograniczenia dla warunków**" określić ograniczenia czasu trwania okna, oraz wprowadzić ograniczenia czasowe użytkownika (rysunek 5.11).
5. Otworzyć okno „**Warunki wystarczające**" (Menu → Planowanie Transportu → Wyznacz warunki), naciskając przycisk „Start" uruchomić proces

wyznaczenia warunków wystarczających (rysunek 5.7). Jeżeli wyznaczony zostanie jeden warunek oznacza to, że odpowiedź na zadane pytania jest pozytywna (Tak), w przypadku braku warunków odpowiedź jest negatywna (Nie).

Dla pytań odpowiadających strukturze przedstawionej w punktach a) i b) wystarczy jednokrotne wyznaczenie warunków wystarczających. Jeśli parametry systemu (struktura połączeń, marszruty, liczba wózków, itp.) nie ulegną zmianie wyznaczony zbiór parametrów może być wielokrotnie używany do udzielania odpowiedzi na pytania zawierające różne rodzaje ograniczeń czasowych i kolejnościowych. Możliwe jest zatem udzielanie odpowiedzi w trybie *on-line*.

W przypadku pytań jak np. przedstawione w punkcie c) wymagane jest każdorazowo wyznaczenie co najmniej jednego warunku wystarczającego. W takich przypadkach system nie gwarantuje wyznaczania odpowiedzi w trybie *on-line*.

5.4.2. Eksperymenty komputerowe

Przeprowadzono szereg badań dotyczących możliwości pracy *SSD* w trybie *on-line*. Polegały one na wyznaczaniu czasu udzielania odpowiedzi, dla wielu wariantów scenariuszy. Pomiar czasu przeprowadzony został dla systemów transportowych o różnej liczbie procesów. Rozważane systemy zawierały od 8 do 25 cyklicznie realizowanych procesów. Systemy charakteryzowały się strukturą nieregularną z różną liczbą zasobów współdzielonych i lokalnych. Realizowane procesy charakteryzowały się długością marszrut od 3 do 7 zasobów.

Ze względu na to, że systemom ze stałą liczbą procesów i stałą liczbą zasobów odpowiadają różne warianty realizacji poszczególnych procesów, wprowadzony został tak zwany współczynnik nasycenia G_T. Określa on, w skali całego systemu transportowego, stopień podziału poszczególnych zasobów pomiędzy realizowane procesy. Definiowany jest on następująco:

$$G_T = \frac{\sum_{i=1}^{q} G_{T,i}}{q}, \qquad (5.1)$$

gdzie: q – liczba realizowanych procesów,

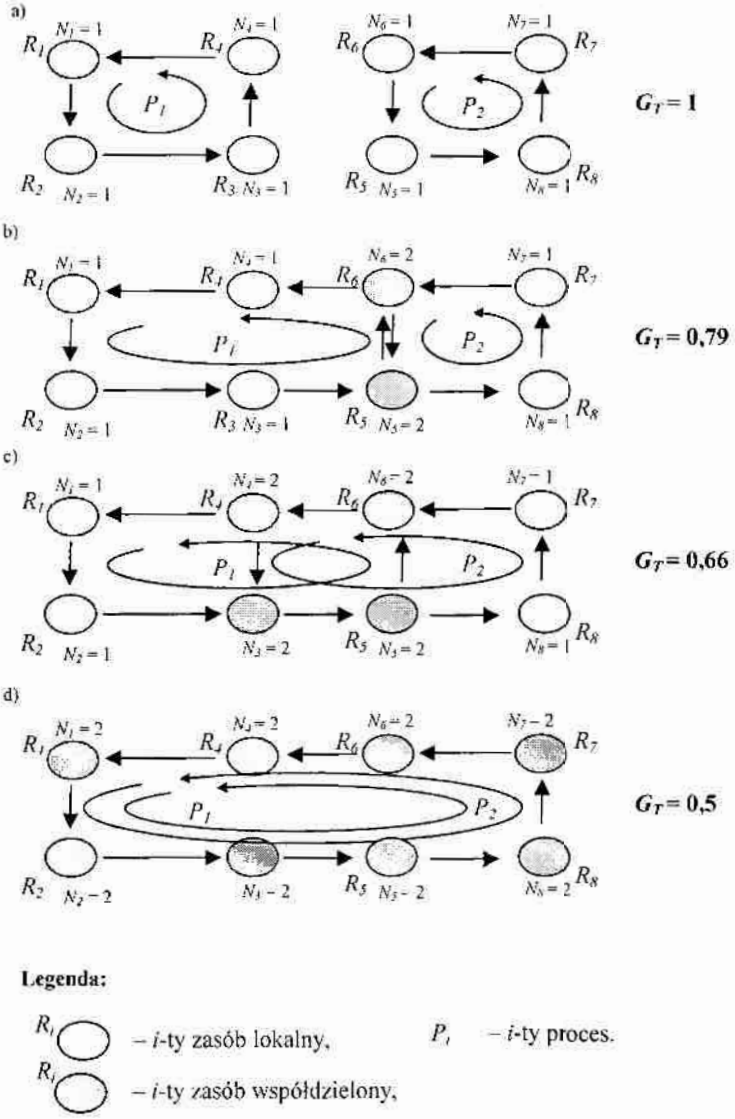
$G_{T,i}$ – współczynnik nasycenia i-tego procesu,

$G_{T,i} = \frac{\sum_{a=1}^{m_i} g_{i,a}}{m_i}$, m_i – długość marszruty i-tego procesu,

$g_{i,a} = \frac{1}{N_j}$, – współczynnik podziału zasobu R_j, R_j – zasób na którym realizowana jest a-ta operacja i tego procesu ,

N_j - liczba procesów realizowanych na zasobie R_j.

Intuicja zaproponowanego współczynnika została zilustrowana na rysunku 5.13. W przypadku gdy procesy są realizowane niezależnie (rysunek 5.13 a)) współczynnik nasycenia $G_T = 1$. W przypadkach gdy procesy są od siebie uzależnione (tzn. realizowane są na wspólnych zasobach) współczynnik $G_T < 1$ (rysunek 5.13 b), 5.13 c), 5.13d)). Wraz ze wzrostem liczby wspólnych zasobów współczynnik nasycenia dąży do zera.



Rys. 5.13. Przykłady systemów z różnymi współczynnikami nasycenia: a) $G_T = 1$, b) $G_T = 0{,}79$, c) $G_T = 0{,}66$, d) $G_T = 0{,}5$

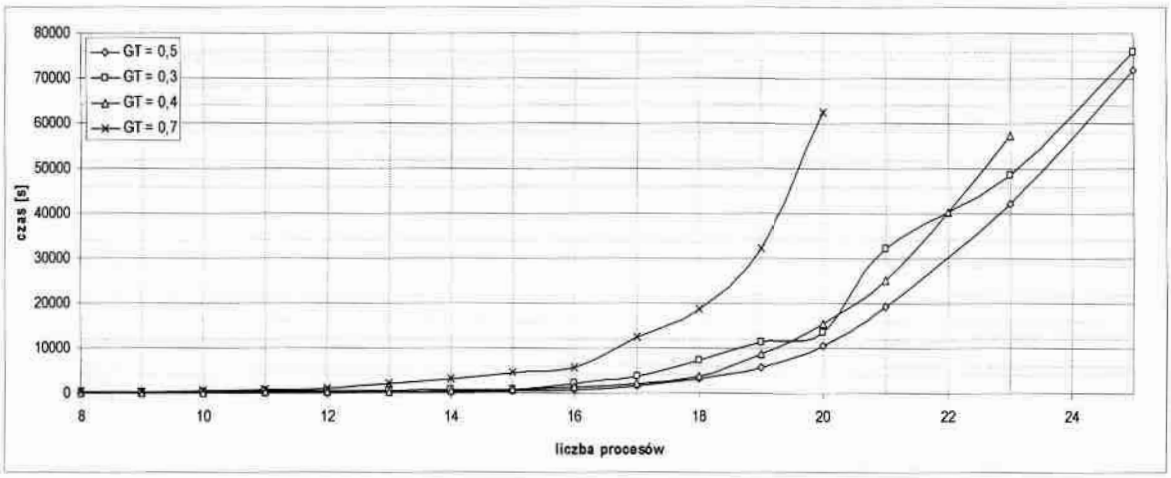
Pierwszym etapem przeprowadzonych badań był pomiar czasów wyznaczania wszystkich warunków wystarczających gwarantujących brak blokady i kolizji dla różnych struktur systemów transportowych.

Pomiar dotyczył czasu generowania reprezentacji wiedzy, czasu kompilacji, i wyznaczania wszystkich alternatywnych warunków wystarczających. Badania przeprowadzono na komputerze z procesorem: Duo T2300E i pamięcią RAM: 512 MB.

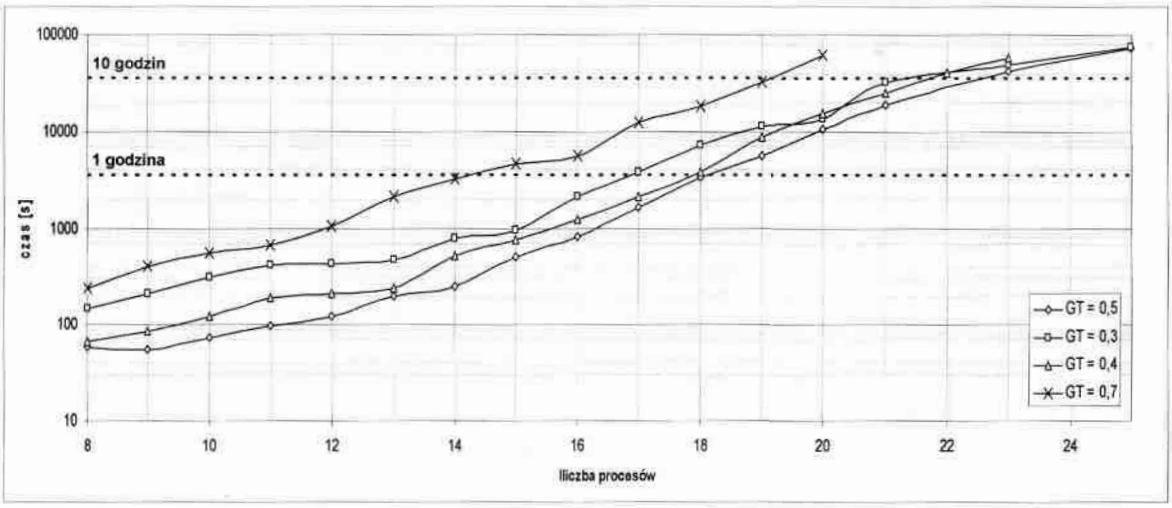
Szczegółowe dane dotyczące przeprowadzonych eksperymentów zawarto w dodatku A. Na rysunku 5.14 przedstawiono zależności czasu wyznaczania odpowiedzi od liczby

realizowanych procesów (od 8 do 25) i współczynnika nasycenia (G_T = 0,3; G_T = 0,4; G_T = 0,5; G_T = 0,7).

a)



b)



Rys. 5.14. Zależność czasu wyznaczenia warunków wystarczających od liczby realizowanych w systemie procesów: a) zależność w skali liniowej, b) zależność w skali logarytmicznej

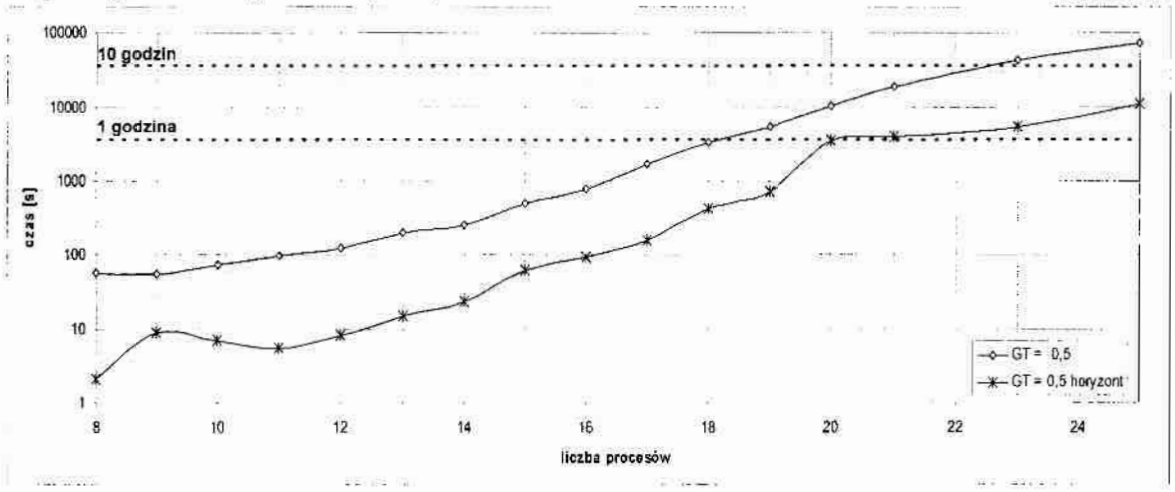
Rysunek 5.14 a) przedstawia czasy wyznaczania wszystkich alternatywnych warunków wystarczających dla systemów transportowych o różnym współczynniku nasycenia. Podczas badań przyjęto założenie, że wyznaczane warunki gwarantują bezblokadową i bezkolizyjną pracę procesów. Przeprowadzone badania odpowiadają zatem realizacji punków 1-3 scenariusza a), gdzie użytkownik nie definiuje żadnych własnych ograniczeń (czasowych i kolejnościowych).

Najmniejszym czasem odpowiedzi charakteryzowały się systemy o współczynniku nasycenia G_T = 0,5, największym czasem systemy o współczynniku 0,7. Sytuację tę można tłumaczyć tym, że wraz ze wzrostem współczynnika nasycenia następuje zmniejszenie liczby zasobów współdzielonych a tym samym zmniejszenie liczby ograniczeń *PSO*. Podczas poszukiwania rozwiązań konieczne jest zatem przeszukiwanie większego obszaru przestrzeni potencjalnych rozwiązań, co jest kosztowniejsze czasowo. Jednak mimo to, dla systemów

transportowych o współczynniku G_T niższym niż 0,5 czas odpowiedzi zaczyna rosnąć (charakterystyki G_T = 0,4; G_T = 0,3). Systemy o niższym współczynniku charakteryzują się większą liczbą ograniczeń i większą liczbą zmiennych (wynikającą z większej liczby zasobów współdzielonych). Wzrost liczby zmiennych powoduje zwiększenie rozmiaru przestrzeni potencjalnych rozwiązań, a tym samym zwiększenie czasu odpowiedzi.

Na rysunku 5.14 b) przedstawione zostały zależności czasowe w skali logarytmicznej. Liniowy charakter otrzymanych wykresów wskazuje na wykładniczą złożoność obliczeniową procesu wyznaczania warunków wystarczających. Na wykresie widać, że w kontekście rozważanych wariantów, zrealizowany *SSD* umożliwia wyznaczenie warunków dla systemów transportowych zwierających od 14 do 18 procesów w czasie jednej godziny. Wyznaczanie warunków dla systemów zawierających do 22 procesów zajmuje do 10 godzin. W oparciu o przeprowadzone eksperymenty można przyjąć, że wzrost liczby procesów o 4 (dla niektórych przypadków o 5) powoduje 10 krotny wzrost nakładów obliczeniowych.

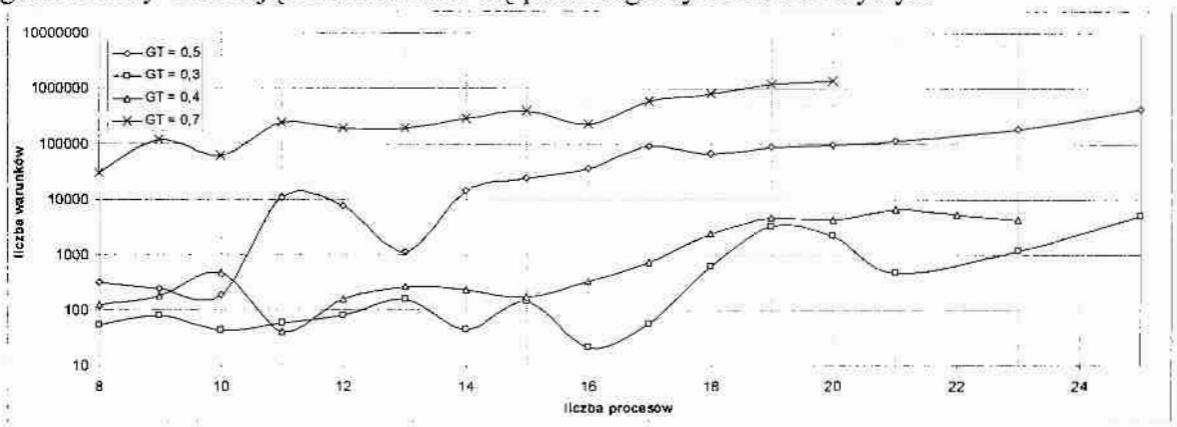
Zwiększenie zakresu stosowania *SSD* jest możliwe poprzez wprowadzanie dodatkowych ograniczeń na etapie wyznaczania warunków wystarczających (punkt 3 scenariusz a)). Celem ilustracji tego faktu przeprowadzony został pomiar czasu wyznaczania warunków wystarczających przy dodatkowo zadanych ograniczeniach wartości czasu trwania cyklu realizacji wszystkich procesów. Na rysunku 5.15, przedstawione zostały dwie charakterystyki czasu wyznaczania warunków wystarczających dla systemów transportowych o współczynniku nasycenia G_T = 0,5.



Rys. 5.15. Zależność czasu wyznaczenia warunków wystarczających od liczby realizowanych w systemie procesów z dodatkowym ograniczeniem czasu trwania cyklu

Charakterystyka „GT = 0,5 horyzont" przedstawia czasy wyznaczania warunków przy uwzględnieniu ograniczeń horyzontu. W większości przypadków otrzymane wyniki są o rząd lepsze niż w przypadku braku ograniczeń (charakterystyka „GT = 0,5"). W ciągu jednej godziny można uzyskać rozwiązania dla systemów realizujących 20 procesów. W rzeczywistości *SSD* pozwala na wyznaczanie warunków wystarczających dla systemów transportowych realizujących do 30 procesów w trybie *off-line*.

Prowadzone badania obejmowały również pomiar liczby warunków wystarczających w zależności od liczby procesów i współczynnika nasycenia. Na rysunku 5.16 przedstawione zostały zależności średniej liczby wyznaczonych warunków dla systemów transportowych charakteryzujących się różnym współczynnikiem nasycenia. Mimo, że liczba warunków wystarczających jest wartością niedeterministyczną (zmiana jednego parametru może spowodować znaczne zwiększenie lub zmniejszenie liczby warunków) i tak można wskazać ogólne trendy określające zachowanie się poszczególnych charakterystyk.



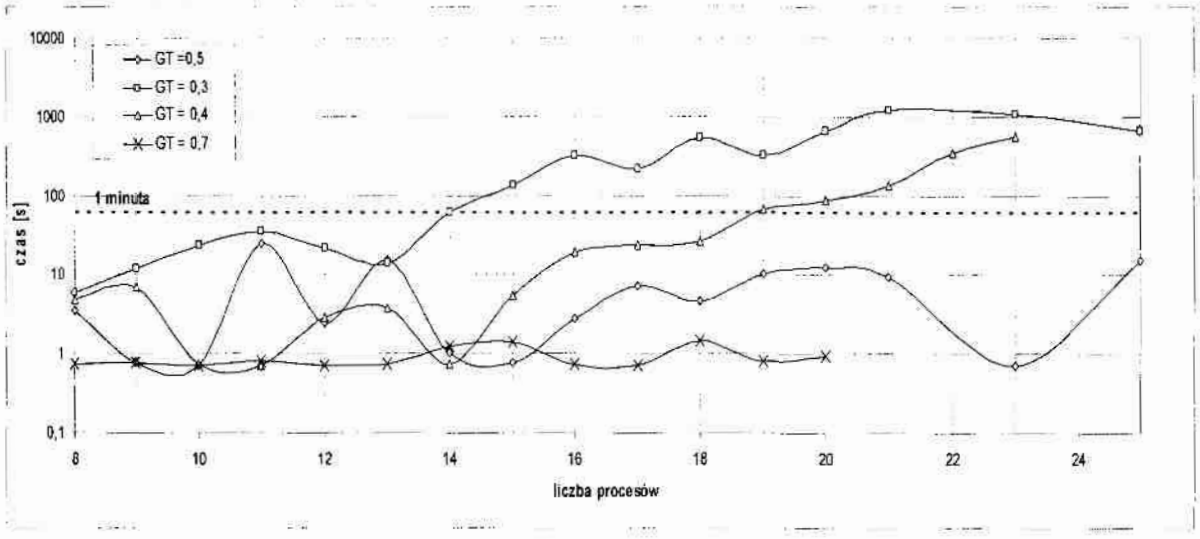
Rys. 5.16. Zależność liczby wyznaczonych warunków od liczby procesów realizowanych w systemie transportowym

Wraz ze wzrostem współczynnika nasycenia rośnie liczba wyznaczonych warunków wystarczających. Dla systemów o współczynniku G_T = 0,7 liczba uzyskiwanych warunków mieści się w granicach od kilkunastu tysięcy do około miliona. W przypadku systemów o współczynniku G_T = 0,3 liczba uzyskiwanych warunków mieści się w granicach od kilku do kilku tysięcy.

Dla systemów o dużym współczynniku nasycenia można zatem zadać górne ograniczenie liczby wyznaczanych warunków, co pozwoli na skrócenie czasu ich wyznaczania. W ten sposób utracona zostaje jednak gwarancja istnienia odpowiedzi na zadane pytanie. W przypadku nie wyznaczenia wszystkich warunków, może powstać sytuacja, w której dla zadanego pytania nie istnieje warunek określający istnienie odpowiedzi. W takim przypadku nie ma gwarancji odpowiedzi zadane pytanie.

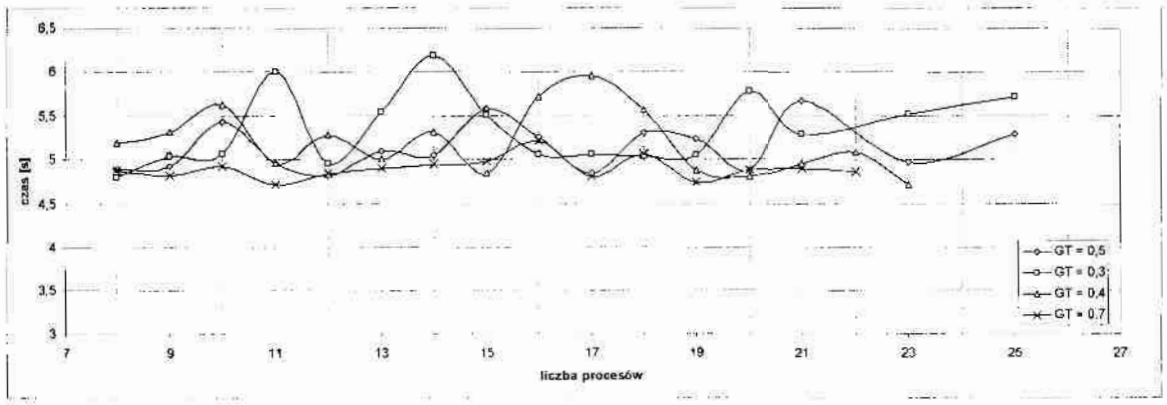
W przypadku realizacji scenariusza c) istotne jest posiadanie wiedzy na temat czasu wyznaczania pierwszego warunku wystarczającego. Istnienie (bądź nie) tego warunku stanowi odpowiedź na pytania dotyczące np. zmiany struktury połączeń, zmiany marszrut, zmiany liczby realizowanych procesów, itp. Przeprowadzone w tym celu pomiary czasu wyznaczania pierwszego warunku wystarczającego w zależności od liczby realizowanych procesów zilustrowane zostały na rysunku 5.17.

Okazuje się, że dla systemów charakteryzujących się współczynnikami nasycenia G_T = 0,7 i G_T = 0,5 czas uzyskania pierwszego warunku wystarczającego nie przekraczał 1 minuty. W takich przypadkach udzielanie odpowiedzi na pytania opisane w scenariuszu c) może być realizowane w trybie *on-line*.



Rys. 5.17. Zależność czasu wyznaczania pierwszego warunku wystarczającego od liczby procesów realizowanych w systemie transportowym

W ogólnym przypadku proces wyznaczania warunków wystarczających jest realizowany w trybie *off-line*. W przypadku pytań odpowiadających scenariuszom a), b) możliwe jest udzielnie odpowiedzi w oparciu o wcześniej wyznaczony zbiór warunków wystarczających. Dla znanych wcześniej zbiorów warunków przeprowadzone zostały pomiary czasu udzielania odpowiedzi na pytania dotyczące harmonogramów spełniających zadane ograniczenia. Wyniki pomiarów zostały przedstawione na rysunku 5.18.



Rys. 5.18. Zależność czasu wyznaczania rozwiązania (harmonogramu) w zależności od liczby procesów

Bez względu na wartość współczynnika nasycenia oraz liczbę procesów czas odpowiedzi mieścił się w granicach 4,5 – 6,5 sekundy. Tak więc w sytuacjach gdy znane są warunki wystarczające odpowiedź udzielana jest w trybie *on-line*.

Podsumowując, *SSD* umożliwia wspomaganie decyzji w systemach transportowych zawierających do 30 procesów. Proces wyznaczania warunków wystarczających charakteryzuje się złożonością wykładniczą i jest realizowany w trybie *off-line*. Proces wyznaczania odpowiedzi na zadane pytania w oparciu o wcześniejsze wyznaczenie warunków wystarczających jest realizowany w trybie *on-line*.

5.5. Porównanie systemu z wybranymi rozwiązaniami

Rozważany w pracy problem stanowi szczególny przypadek sterowania procesami w systemach współbieżnych procesów cyklicznych. W takich systemach, do wyznaczania harmonogramów realizacji poszczególnych operacji, wykorzystywany jest formalizm algebry (max, +). Formalizm algebry (max, +) jest naturalnym formalizmem w modelowaniu systemów, w których współpraca procesów jest oparta na protokole wzajemnego wykluczania. W pracach [50], [51], [52], pokazano możliwość wykorzystania tego typu formalizmu do wyznaczania wielu charakterystyk funkcjonowania systemu, m.in. takich jak harmonogramy przebiegu procesów czy okresy pracy systemu. Metoda wyznaczania charakterystyk funkcjonowania systemów współbieżnych procesów cyklicznych obejmuje etapy: specyfikacji parametrów wejściowych (zmienne opisujące obiekt, struktura systemu), określenia stanu początkowego i reguł priorytetowania, budowy modelu analitycznego (wykorzystującego formalizm (max, +)) i oceny jakości funkcjonowania. Poszukiwanie harmonogramów spełniających wymagania stawiane przez użytkownika (na określonym przez niego poziomie jakości), polega na przyjęciu postaci stanu początkowego, reguł priorytetowania i w oparciu o nie wyznaczeniu parametrów przebiegu procesów (w postaci harmonogramów). W przypadku niepowodzenia, ponownie określane są reguły priorytetowania i stan początkowy, poczym wyznaczany jest harmonogram. Proces ten powtarzany jest aż do uzyskania harmonogramów na żądanym poziomie jakości.

Wadą metod opartych na formalizmie algebry (max, +) jest brak gwarancji wyznaczenia rozwiązania w trybie *on-line*. Do wyznaczenia i oceny harmonogramu konieczne jest posiadanie warunków, w postaci stanu początkowego i reguł priorytetowania a te są wyznaczane metodą prób i błędów. Jednak w przeciwieństwie do zrealizowanego *Systemu Sterowania Dyspozytorskiego* podejście oparte o algebrę (max, +) umożliwia analizę systemów o znacznie większych rozmiarach (liczba procesów > 30). Dodatkową zaletą tego podejścia są znacznie efektywniejsze metody kompresji struktur *SWPC* co umożliwia zwiększenie zakresu stosowania systemów opartych o ten formalizm.

Wad formalizmu algebry (max, +) nie zawiera podejście opracowane przez R. Wójcika przedstawione w pracach [88], [90] [87], [91]. W pracach tych opisane zostały twierdzenia umożliwiające wyznaczenie w trybie *on-line*, bezkolizyjnych i bezblokadowych harmonogramów realizacji poszczególnych procesów. Zaletą tego podejścia jest możliwość budowy systemów wspomagania dla znaczenie większych systemów transportowych niż prezentowany w pracy System Sterowania Dyspozytorskiego. Cechą charakterystyczną jest to, że wyznaczone harmonogramy gwarantują realizację poszczególnych procesów bez oczekiwania na dostęp do zasobów współdzielonych. Harmonogramy charakteryzują się cyklem o wartości równej najmniejszej wspólnej wielokrotności czasów trwania marszrut realizowanych procesów [88]. Jednak podobnie jak w przypadku algebry (max, +) metoda ta nie daje gwarancji istnienia rozwiązania. Na przykład, gdy nie jest możliwe wyznaczenie harmonogramu bez uniknięcia oczekiwania procesów, powyższa metoda nie umożliwia wyznaczenia alternatywnych (kosztowniejszych czasowo) rozwiązań. Stosowanie tego

podejścia jest więc ograniczone w swym zakresie tylko do wyznaczania harmonogramów charakteryzujących się brakiem oczekiwania na dostęp do zasobów współdzielonych.

W kontekście przedstawionych rozwiązań *SSD* stanowi swego rodzaju alternatywę, umożliwiającą wyznaczenie warunków wystarczających oraz harmonogramów w trybie *on-line* bez względu na rodzaj ograniczeń stawianych przez użytkownika. System umożliwia wspomaganie decyzji z zakresu obu powyżej przedstawionych podejść (np. umożliwia wyznaczanie harmonogramów bez oczekiwania), ponadto może być on stosowany wszędzie tam gdzie powyżej opisane metody zawodzą. W przeciwieństwie do przedstawionych metod zawsze daje gwarancję istnienia odpowiedzi, która jest wyznaczana w trybie na bieżąco. Zakres stosowania *SSD* ograniczony jest jednak do mniejszych struktur systemów transportowych niż omówione powyżej podejścia.

5.6. Podsumowanie

Przedstawiona metodyka projektowania systemów interakcyjnego wspomagania decyzji umożliwia budowę pakietów dedykowanych do rozwiązywania określonych klas problemów. Obejmuje ona następujące etapy: specyfikację problemu, wybór języka *CP*, dobór narzędzi *CP*, poszukiwanie warunków wystarczających oraz dobór strategii przeszukiwania przestrzeni rozwiązań. Metodyka ta pozwala na poszukiwanie rozwiązania dowolnego problemu decyzyjnego w oparciu o technologię *CP*. Pozwala ona również na budowanie pakietów umożliwiających pracę z użytkownikiem w trybie na bieżąco.

W oparciu o opracowaną metodykę zbudowano *System Sterowania Dyspozytorskiego*, który wspomaga decyzje dyspozytora podsystemów transportowych.

Wyniki przeprowadzonych eksperymentów wskazują na efektywność zaimplementowanych metod *CP*. Opracowany *SSD* spełnia wymagania systemów interakcyjnego wspomagania decyzji, gwarantuje jednocześnie istnienie odpowiedzi na zadany zbiór rutynowych pytań.

W dodatku B przedstawiono pełny opis opracowanego systemu. W dodatku A przedstawiono szczegółowy opis przeprowadzonych eksperymentów.

6. Zakończenie

6.1. Rezultaty poznawcze

Celem pracy było opracowanie metodyki projektowania zadaniowo zorientowanych systemów wspomagania decyzji funkcjonujących w trybie *on-line*. W szczególności zaś opracowanie systemu wspomagania sterowania dyspozytorskiego w podsystemach transportowych elastycznych systemów produkcyjnych.

Przeprowadzone badania potwierdziły słuszność przyjętej tezy zakładającej, że implementacja metody logiczno-algebraicznej w technikach programowania z ograniczeniami umożliwia budowę systemów interakcyjnego wspomagania decyzji. Przyjęta teza wykazana została dla systemów modelowanych w kategoriach reprezentacji wiedzy. Zastosowany formalizm pozwolił specyfikować obiekty w postaci bazy wiedzy (zbioru faktów i formuł elementarnych), tzn. języka rachunku zdań. Formalizm ten pozwala uniezależnić się od niedostatków powszechnie stosowanych mechanizmów wnioskowania takich jak *modus ponens* (wymagających regułowej bazy wiedzy) czy też zasady rezolucji (wymagającej reprezentacji wiedzy wykorzystującej klauzule, np. klauzule Horna). Przyjęta metoda logiczno-algebraiczna wykorzystuje mechanizm wnioskowania polegający na przeszukiwaniu przestrzeni potencjalnych wartości logicznych formuł elementarnych pod kątem istnienia związków logicznych (w postaci implikacji $Fu(u) \Rightarrow Fy(y)$) między faktami wejściowymi $Fu(u)$ i wyjściowymi $Fy(y)$ bazy wiedzy.

Przyjęty mechanizm wnioskowania w sposób naturalny daje się implementować w technikach programowania z ograniczeniami, gdzie zdania logiczne są traktowane jako ograniczenia, a mechanizmy propagacji ograniczeń i dystrybucji zmiennych decyzyjnych traktowane są jako mechanizmy wnioskowania.

Wymienione zalety modelu pozwoliły rozstrzygnąć problem budowy systemu interakcyjnego wspomagania decyzji. Problem ten dekomponował się na dwa problemy.

Pierwszy z tych problemów dotyczył gwarancji istnienia odpowiedzi na każde ze zbioru postulowanych pytań rutynowych.

Drugi problem wiązał się z gwarancją uzyskania odpowiedzi w trybie *on-line* (w czasie do 5 minut) dla każdego z wariantów postulowanego wcześniej zbioru pytań rutynowych.

Rozwiązanie powyższych problemów znalazło wyraz w przyjętej metodologii budowy systemów wspomagania decyzji obejmującej dwie fazy: wyznaczania warunków wystarczających gwarantujących istnienie odpowiedzi na zadane pytanie oraz gwarancji, że przyjęta procedura rozwiązywania umożliwia pracę w trybie interakcyjnym dla zadań o skali występujących w praktyce. Realizacja wymienionych faz umożliwia wyznaczenie odpowiedzi na problemy, zadanej skali, w trybie na bieżąco.

Rozwiązanie pierwszego z wyżej wymienionych problemów (wyznaczania warunków wystarczających) wymagało:

- wyznaczenia warunków (w rozważanym przypadku systemu transportowego) jakościowych, spełnienie których gwarantuje istnienie odpowiedzi na zadany zbiór pytań (warunków bezblokadowej pracy wózków),
- wyznaczenia warunków ilościowych – określenia ustalonej postaci warunków wystarczających spełnienie, których gwarantuje istnienie ilościowej odpowiedzi na poszczególne pytania rutynowe (w rozważanym przypadku postacią taką były pary typu: (stan początkowy S_0, zbiór reguł priorytetowania Θ), a poszukiwanymi wielkościami były: cykl systemu, harmonogram pracy wózków samojezdnych, itp.)

Rozwiązanie pierwszego podproblemu (wyznaczanie warunków jakościowych) znalazło miejsce we własnościach wyprowadzonych w postaci twierdzeń (Twierdzenie 4.1) i lematów (Lematy 4.1, 4.2, 4.3 i 4.4). Zdobyta w ten sposób wiedza pozwoliła uzupełnić pierwotnie dostępną, aprioryczną bazę wiedzy, specyfikującą rozważany obiekt, o własności dotyczące bezblokadowej i bezkolizyjnej pracy systemu.

Rozwiązanie drugiego podproblemu polegało na zweryfikowaniu spójności rozważanej (poprzez rozwiązanie problemu decyzyjnego) wiedzy, to znaczy do odpowiedzi na pytanie: Czy w przypadku każdego pytania rutynowego (dotyczącego harmonogramowania) istnieją alternatywne warunki spełnienie, których gwarantuje odpowiedź na pytanie.
Oznacza to, że w każdym przypadku, każdego pytania rutynowego, sprawdzane było istnienie bądź nie, związku przyczynowego pomiędzy instancjami zmiennych decyzyjnych odpowiadającymi właściwościom wejściowym $Fu(u)$ i wyjściowym $Fy(y)$ bazy wiedzy.
Wykorzystane wnioskowanie odpowiada wnioskowaniu wstecznemu w systemach z regułową bazą wiedzy. W przyjętych technikach programowania z ograniczeniami podczas wnioskowania „gubiony" jest łańcuch wnioskowań pośrednich.

Rozwiązanie drugiego z rozważanych problemów (związanego z gwarancją udzielania odpowiedzi na zadane pytania w trybie interakcyjnym) wiązało się z rozwiązaniem następujących dwóch podproblemów:

- kompresji danych umożliwiającej bezstratne odtwarzanie szczegółowej wiedzy o strukturze i zachowaniu modelowanego systemu (w rozważanym przypadku sparametryzowanej postaci faktów opisującej funkcjonowanie sytemu transportowego oraz budowy stref zasobów współdzielonych i lokalnych występujących w poszczególnych marszrutach transportowych).
- wyznaczenia efektywnych czasowo strategii przeszukiwania (w rozważanym przypadku dwuetapowej strategii przeszukiwania rozwiązań dedykowanych).

Rozwiązanie pierwszego z podproblemów sprowadzało się do wyznaczania tych podzbiorów faktów rozważanej bazy wiedzy, które pozwalają się sprowadzić do jednego faktu sparametryzowanego (zbiór sparametryzowanych faktów tworzy schemat faktów). Przyjęcie wprowadzonego schematu faktów przyspiesza proces budowy bazy wiedzy specyfikującej badany obiekt oraz przyspiesza poszukiwanie odpowiedzi poprzez zmniejszenie przestrzeni potencjalnych rozwiązań.

Koncepcja zastępowania pewnego obszaru zasobów jednego typu poprzez zasoby zastępcze, powoduje zmniejszenie liczby faktów i zmiennych decyzyjnych koniecznych do specyfikowania problemu. Prowadzi to do zmniejszenia przestrzeni potencjalnych zmiennych, a w konsekwencji pozwala przyspieszyć jej przeszukiwanie.

W celu rozwiązania drugiego podproblemu związanego z wyznaczeniem strategii efektywnego czasowo przeszukiwania, wykorzystuje się fakt, że dla rozwiązywanego problemu decyzyjnego istotne są tylko wartości niektórych zmiennych decyzyjnych. W szczególności oznacza to, że w przypadku przestrzeni trzech zmiennych decyzyjnych (ewentualnie trzech zbiorów zmiennych) – zmiennych wejściowych u, wewnętrznych w i wyjściowych y – dystrybucja tylko zmiennych wejściowych pozwala ograniczyć się do przeszukiwania fragmentów drzewa potencjalnych rozwiązań. Częściowe przeszukiwanie drzewa przyspiesza otrzymanie rozwiązania

Rozwiązanie powyższych problemów znalazło swoje zastosowanie w schemacie metodyki budowy systemów interakcyjnego wspomagania decyzji, w szczególności rozwiązywanie w trybie *off-line* wyżej przedstawionych problemów stanowi swoistą „skorupę" poszukiwanego systemu wspomagania.

Przedstawione rozwiązania dostarczają zbiór warunków wzbogacających bazę wiedzy modelowanego systemu gwarantujących istnienie rozwiązania oraz mechanizmy wnioskowania wyrażające się w efektywnej czasowo (implementowanej w technikach programowania z ograniczeniami) strategii propagacji i dystrybucji zmiennych.

6.2. Rezultaty utylitarne

W ramach pracy opracowany został *System Sterowania Dyspozytorskiego* dedykowany dla dyspozytorów podsystemów transportowych elastycznych systemów wytwarzania. Ze względu na zastosowanie systemu klasy Open Source (**Oz Mozart**) oraz języka programowania *Java*, system ten spełnia wymagania finansowe potencjalnych odbiorców, np. małych i średnich firm. Zastosowanie „przyjaznego" interfejsu użytkownika eliminuje potrzebę zatrudniania wysokokwalifikowanej kadry.

Budowa systemu obejmowała realizację dwóch podstawowych modułów:

- modułu *off-line* (w rozważanym przypadku jest to moduł wyznaczania warunków wystarczających),
- modułu *on-line* (w rozważanym przypadku jest to moduł wyznaczania odpowiedzi w oparciu o warunki wystarczające zadane w postaci zbioru stanów początkowych i reguł priorytetowania).

Pierwszy z modułów (*off-line*) implementuje rozwiązania umożliwiające wyznaczanie warunków wystarczających dla zadanych parametrów systemu transportowego. Pozwala on na wyznaczanie warunków ilościowych spełniających zadane przez użytkownika systemu ograniczenia. Podstawowe funkcje modułu to generowanie reprezentacji wiedzy, w oparciu o przygotowaną postać schematu faktów oraz rozwiązywanie problemu decyzyjnego

(w środowisku programowania z ograniczeniami **Oz Mozart**) przy użyciu dwuetapowej strategii przeszukiwania. Poszukiwanie warunków wystarczających odbywa się w trybie *off-line*.

Drugi z modułów (*on-line*) implementuje rozwiązania umożliwiające wyznaczanie odpowiedzi systemu (w postaci gotowych harmonogramów) na zadane przez użytkownika pytania. Podstawową funkcją modułu jest rozwiązywanie problemu spełniania ograniczeń (w trybie *on-line*), zawierającego (w postaci ograniczeń) wyznaczone warunki wystarczające, przy użyciu dwuetapowej strategii przeszukiwania.

Modułowa budowa umożliwia rozdzielenie systemu na dwie niezależnie aplikacje (jedną odpowiedzialną za wyznaczanie warunków wystarczających, drugą odpowiedzialną za poszukiwanie odpowiedzi na zadane pytania w trybie interakcyjnym). W opracowanym systemie interfejs użytkownika jest odseparowany od modułu obliczeniowego co umożliwia to jego łatwą modyfikację. Sam moduł obliczeniowy (głównie moduł *on-line*) może też być wykorzystany jako część składowa (odpowiedzialna za harmonogramowanie pracy procesów) większego systemu wspomagania decyzji, np. systemu sterowania operacyjnego.

Główne funkcje opracowanej aplikacji pozwalają na: generowanie/weryfikację harmonogramów pracy wózków samojezdnych w kontekście zadanej struktury tras systemu transportowego, ograniczeń użytkownika, parametrów systemu transportowego oraz założeń wynikających z definiowanej klasy systemów.

Przeprowadzone eksperymenty wykazały, że system spełnia wymagania stawiane systemom interakcyjnego wspomagania decyzji. System umożliwia (moduł *off-line*) generowanie warunków wystarczających dla systemów transportowych obsługujących do 30 wózków samojezdnych, charakteryzowanych współczynnikiem nasycenia z zakresu: $0,3 \leq G_T \leq 0,7$. W oparciu o uprzednio wyznaczone warunki wystarczające, moduł *on-line* umożliwia wyznaczenie odpowiedzi na zadane pytanie w czasie poniżej 10 sekund.

Dla problemów rozważanej skali możliwa jest zatem praca systemu w trybie interakcyjnym.

6.3. Kierunki dalszych badań

Jako główne kierunki dalszych badań należy wskazać: rozszerzenie przedstawionej metodyki o możliwość projektowania systemów wspomagania decyzji dla problemów o charakterze rozmytym, połączenie technik programowania z ograniczeniami z metodami dekompozycji, które obecnie są wykorzystywane do rozwiązywania problemów metody logiczno-algebraicznej oraz poszukiwanie efektywniejszych metod kompresji reprezentacji wiedzy.

W pracy skupiono się głównie na budowie systemów wspomagania decyzji dla obiektów opisywanych za pomocą zmiennych ostrych. Ważnym obszarem dalszych badań jest rozszerzenie przedstawionego podejścia dla obiektów charakteryzowanych zmiennymi rozmytymi. Formułowanie specyfikacji określonego problemu w postaci rozmytej wymaga przeniesienia metod programowania z ograniczeniami i procedur wyznaczania warunków

wystarczających na płaszczyznę logiki rozmytej. Z punktu widzenia logiki rozmytej istotne jest posiadanie metody umożliwiającej rozwiązywanie problemu decyzyjnego (wnioskowanie wstecz), przy użyciu klasycznych technik *CP*. Pierwsze prace w tym kierunku zostały już podjęte [2], [3], [4].

Kierunkiem rozwoju zaproponowanej strategii jest próba połączenia technik programowania z ograniczeniami z metodami dekompozycji (wykorzystujące algorytmy rekurencyjne) prezentowanymi w pracach [38], [28]. Reprezentacja wiedzy jest dekomponowana na określoną liczbę elementów (cząstkowe reprezentacje wiedzy). W takim przypadku rozwiązanie problemu decyzyjnego polegałoby na rozwiązaniu wielu oddzielnych problemów cząstkowych *PSO*. Każdy z dekomponowanych problemów charakteryzuje się mniejszą liczbą zmiennych i ograniczeń przez co może być rozwiązywany szybciej. Powstaje więc pytanie czy sumaryczny czas rozwiązania wszystkich podproblemów jest mniejszy niż w przypadku nie zdekomponowanego problemu *PSO*? Można postawić hipotezę, że tego typu podejście, łączące zalety dwóch niezależnych metod (metod programowania z ograniczeniami i metod dekomopozycji), może być w stosunku do nich atrakcyjne pod kątem czasu otrzymania rozwiązań.

W kontekście poruszonego w pracy problemu harmonogramowania pracy wózków samojezdnych, proponowany kierunek dalszych prac dotyczy metod kompresji schematu faktów. Wiąże się to z zagadnieniami kompresji struktur transportowych charakteryzujących się pewnymi regularnościami, na przykład kompresji struktur segmentowych (składających się ze skończonej liczby identycznych podstruktur). Istnieją opracowania [52], wskazujące na możliwość realizacji tego typu kompresji w przypadku stosowania algebry (max, +). Powstaje pytanie czy tego typu kompresje mogą być stosowane w przypadku przedstawionego podejścia?

Literatura

[1] Arbib A.M, Bobrow L.S.: *Discrete Mathematics*, W.B Saunders Company, Philadelphia, 1974.

[2] Bach I., Tomczuk-Piróg I., Relich M.: *Metody modelowania rozmytego w ocenie harmonogramowania produkcji*, Technologiczne Systemy Informacyjne w Inżynierii Produkcji i Współczesne Technologie w Budowie Maszyn, Kazimierz Dolny, 2006, str. 28 – 32.

[3] Bach I., Ważna L., **Bocewicz G.**: *Wykorzystanie modelowania rozmytego i technik CP/CLP do budowy narzędzia wspomagania programowania inwestycji*, Inżynieria Produkcji, Zielona Góra, 2006, str. 7 – 31.

[4] Bach I., Tomczuk-Piróg I., Banaszak Z.: *Zastosowanie technik CLP do oceny efektywności przedsięwzięć inwestycyjnych w warunkach niepewności,* Konferencja Zarządzanie Przedsięwzięciem - teoria i praktyka, AGH, Kraków, 2006, str. 243 - 249

[5] Banaszak Z., **Bocewicz G.**, Bzdyra K.: *Zastosowanie metod programowania z ograniczeniami do planowania zajęć*, VI Międzynarodowa KN-T, Technologiczne Systemy Informacyjne w Inżynierii Produkcji i Kształceniu Technicznym, Kazimierz Dolny, 2005, str. 20 - 27.

[6] Banaszak Z, **Bocewicz G.**: *Towards unified framework for dedicated DSS design*, Materiały KN-T Automatyzacja - Nowości i Perspektywy, PIAP, Warszawa, 2006, pp. 298 – 307.

[7] Banaszak Z., **Bocewicz G.**: *CLP based decision support system methodology for SME management,* Seminarium nt.: Modelowanie i optymalizacja procesów produkcyjnych, zachodnio czeski Uniwersytet w Pilźnie, Republika Czeska, 2006, pp. 15 – 21.

[8] Barták R., *Constraint programming: what is behind?*, Proceedings of the Workshop on Constraint Programming for Decision and Control, 1999, pp. 7-15.

[9] Barták R., *Incomplete Depth-First Search Techniques: A Short Survey*, Proceedings of the 6th Workshop on Constraint Programming for Decision and Control, Ed. Figwer J., 2004, pp. 7-14.

[10] Barták R.: *Constraint programming: In pursuit of the holy grail,* http://kti.mff.cuni.cz/~bartak/constraints/", 2005.

[11] Ben-Ari M.: *Logika matematyczna w informatyce*, WNT, Warszawa, 2005.

[12] Białko M.: *Podstawowe właściwości sieci neuronowych i hybrydowych systemów ekspertowych*, Wydawnictwo Uczelniane Politechniki Koszalińskiej, 2000.

[13] Białko, M.: Condensed CLIPS 5.1 User's Guide, Politechnika Koszalińska,1995.

[14] Błażewicz J.: *Złożoność obliczeniowa problemów kombinatorycznych.* WNT, Warszawa, 1988.

[15] **Bocewicz G.**: *Zastosowanie metody logiczno algebraicznej i technik programowania z ograniczeniami do badania poprawności wiedzy,* Modele inżynierii teleinformatyki Politechnika Koszalińska 2006, pp. 21-27.

[16] **Bocewicz G.**, Bach I., Banaszak Z., Jakubowski J.: *Zastosowanie technik programowania w logice z ograniczeniami do wyznaczania drzew diagnostycznych.* Przegląd Mechaniczny, Rok wyd. LXV, Zeszyt 12S/2006, str. 28-31.

[17] **Bocewicz G.**, Banaszak Z.: *Logic-algebraic method based and CP driven approach to SMEs Knowledge Base consistency checking*, Computer Science Driven Production Engineering, University of Zielona Góra, 2006, pp. 7-26.

[18] **Bocewicz G.**, Banaszak Z.: *Zastosowania technik programowania z ograniczeniami w systemach wspomagania sterowania dyspozytorskiego procesami transportu i składowania ESP*, Efektywność obrabiarek i procesów produkcyjnych, Inżynieria Maszyn Zeszyt 1, 2006, pp. 100 – 111.

[19] **Bocewicz G.**, Banaszak Z., Bzdyra K.: *Zastosowanie metody logiczno-algebraicznej do badania własności systemów współbieżnych procesów cyklicznych.* Zarządzanie i Inżynieria Produkcji, 2005, Bielsko Biała, str. 45 – 55.

[20] **Bocewicz G.**, Banaszak Z., Wójcik R.: *Design of systems of concurrently competing cyclic processes: A logic-algebraic method approach.* AI-METH – Artificial Intelligence Methods, Gliwice, 2005, pp. 93 – 98.

[21] **Bocewicz G.**, Bzdyra K.: *Graficzna metoda planowania zajęć*, VI Międzynarodowa KN-T, Technologiczne Systemy Informacyjne w Inżynierii Produkcji i Kształceniu Technicznym, Kazimierz Dolny, 2005, str. 27 – 34.

[22] **Bocewicz G.**, Muszyński W., Banaszak Z.: *Zastosowanie technik programowania z ograniczeniami do budowy interakcyjnego systemu sterowania dyspozytorskiego procesami transportu i składowania ESP.* W: Metody i techniki zarządzania w inżynierii produkcji. Matuszek J. (Red.), Wydawnictwo Akademii Techniczno-Humanistycznej, Bielsko-Biała, 2006, str. 107–126.

[23] **Bocewicz G.**, Banaszak Z., Wójcik R.: *Design of admissible schedules for AGV systems with constraints: a logic-algebraic approach.* The 1st KES Symposium on Agent and Multi-Agent Systems – Technologies and Applications, Wrocław 2007, (w druku).

[24] Borowiecki T.: *Synteza modeli procedur planowania realizacji zleceń produkcyjnych,* praca doktorska, Politechnika Poznańska 2006.

[25] Bubnicki Z.: *Wstęp do systemów ekspertowych*, PWN, Warszawa, 1990.

[26] Bubnicki Z.: *Procesy uczenia i metoda logiczno-algebraiczna w systemach z reprezentacją wiedzy*, Analiza systemowa i zarządzanie, Inst. Badań Sys.PAN wydanie: I, 1999.

[27] Bubnicki Z.: *Logic-algebraic method for a class of knowledge based systems,* F. Picher, R. Moreno Diaz (red.). Computer Aided Systems Theory. Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1333, 1997.

[28] Bubnicki Z.: *Logic-algebraic method for knowledge-based relation systems,* Systems Analysis Modeling and Simulations, vol. 33, 1998.

[29] Bzdyra K.: *Algorytmy planowania przepływu produkcji wykorzystujące techniki programowania w logice ograniczeń*, praca doktorska, Politechnika Koszalińska, 2005.

[30] Bzdyra K., Banaszak Z.: *Planowanie przebiegu procesów produkcyjnych z wykorzystaniem technik programowania z ograniczeniami*, Komputerowo Zintegrowane Zarządzanie, pod red. R. Knosali, WNT, Warszawa, 2004, str.165 –171.

[31] Bzdyra K., Tomczuk I., Banaszak Z.: *Modelowanie procesów dystrybucji w języku Mozart*, Inżynieria Maszyn vol. 8. Zeszyt 2, Wrocław, 2003, str. 68 – 75.

[32] Bzdyra K., Tomczuk I., Banaszak Z.: *Production flow planning based on a reference model of constraint satisfaction problem decomposition*, Materiały KN-T Automatyzacja - Nowości i Perspektywy, PIAP Warszawa, 2005, str. 233 – 242.

[33] Chan P., Heus K., Veil G.: *Nurse scheduling with global constraints in CHIP: Gymnaste*, In Proc. PACT98, 1998.

[34] Chrobot J. Rakowski J.: *Systemy operacyjnego sterowania wytwarzaniem*, praca doktorska, Politechnika Wrocławska, 1998.

[35] Clocksin W. F., Mellish C. S.: Prolog. Programowanie, Helion 2003.

[36] Dubois D., Fargier H., Fortemps P.: *Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge,* European Journal of Operational Research 147, 2003, pp. 231 – 252.

[37] Freuder E.: (ed.) *Principles and Practice of Constraint Programming-CP96*, Springer-Verlag, LNCS 1118, 1996.

[38] Fraś M.: *Metodyka i algorytmy obliczeniowe w systemach wieloprocesowych dla metody logiczno-algebraicznej,* praca doktorska, Politechnika Wrocławska, 2004.

[39] Gilbert D.R.: *Bioinformatics and constraint (logic) programming*, Logic Programming Newsletter, 1998.

[40] Glaisner F., Richard L.: *FORWARD-C: A refinery scheduling system.* In Proc. PACT97, 1997.

[41] Graham P.: *On Lisp, advanced techniques for common Lisp*, Prentice Hall, 1993.

[42] Grzegorczyk A.: *Zarys logiki matematycznej*, PWN, Warszawa, 1973

[43] Hendry L.C., Kingsman B.G.: *Production planning systems and their applicability to make to order companies*, European Journal of Operational Research, Vol. 40, 1989, pp. 1-15.

[44] Janiak A.: *Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów*, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1999.

[45] Kokkoras F., Gregory S.: *D-WMS: Distributed workforce management using CLP*, In Proc. PACT98, 1998, pp. 129-146.

[46] Lawley M.A., Reveliotis S.A., Ferreira P.M.: *A correct and scalable deadlock avoidance policy for flexible manufacturing systems*, IEEE Trans. on Robotics and Automation, Vol.14, No.5, 1998, str. 796 – 809.

[47] Legierski W., Parys P.: *Planowanie zajęć metodami programowania z ograniczeniami*, Automatyzacja Procesów Dyskretnych, tom: Optymalizacja Dyskretna. Robotyka i sterowniki programowalne, WNT,Warszawa, 2004, str. 125 – 132.

[48] Ligęza A.: *Logical foundations for rule-based systems*, AGH, Kraków , 2005.

[49] Luo X., Ho-man Leeb J., Ho-fung Leung, Jennings N.R.: *Prioritised fuzzy constraint satisfaction problems: axioms, instantiation and validation*, Fuzzy Sets and Systems 136, 2003, pp. 151–188.

[50] Majdzik P., Banaszak Z.: *Procedura sterowania rozproszonego w systemach współbieżnych procesów cyklicznych*, Modele algorytmy i standardy w informatyce, Zeszyty Naukowe, WSKiZ Poznań, 2005, str. 19 – 38.

[51] Majdzik P., Polak M., Wójcik R., Banaszak Z.: *Effektive prototyping of concurrent processes systems with multiple dispatching rules*, Proccedings of the CS&P 2005 : workshop. Ruciane-Nida, Polska, Warsaw, 2005.

[52] Majdzik P., Banszak Z., *Warunki kompozycji systemów współbieżnych procesów cyklicznych*, Zeszyty Naukowe PŚl, seria: AUTOMATYKA, Nr 1726, Zeszyt, nr 143, Gliwice, 2006, str. 91 – 98.

[53] Mądry K.: *Planowanie obsługi sieci dystrybucji odpadów poprodukcyjnych z wykorzystaniem technik programowania ograniczeń*, praca doktorska, Politechnika Poznańska, 2006.

[54] Mądry M., Saniuk S., Banaszak Z.: *Prototypowanie zleceń produkcyjnych w jednoczesnej produkcji wieloasortymentowej*, Materiały konferencyjne VII Międzynarodowej konferencji naukowej Zarządzanie organizacjami gospodarczymi w warunkach globalizacji, Łódź, 2000, str. 435 – 446.

[55] Mądry M., Saniuk S., Banaszak Z.: *Zastosowanie metod programowania w logice ograniczeń do planowania przepływu w sieciach dystrybucji*, Komputerowo zintegrowane zarządzanie T. 2, red. E. Knosala, Warszawa, 2005, str. 121– 130.

[56] Meier M., Schimpf J.: *An Architecture for Prolog Extensions*, Proceedings of 3rd International Workshop on Extensions of Logic Programming, Bologna, 1992.

[57] Miłosz M.: *Solver Lingo w rozwiązywaniu dużych modeli optymalizacyjnych.* W: Zarządzanie przedsiębiorstwem – ekonomia, prawo, kultura, etyka. Pod red. Wł. Sitko. Kazimierz Dolny, 2001, str. 23– 33.

[58] Mulawka J.J.: *Systemy ekspertowe*, WNT, Warszawa 1996.

[59] Muszyński W., **Bocewicz G.**, Banaszak Z.: *Harmonogramowanie pracy wózków samojezdnych w warunkach ograniczonego dostępu do zasobów współdzielonych ESP (Problem decyzyjny)*, IX Konferencja Automatyki Robotyki, KKR, Piechowice, 2006, str. 175 – 187.

[60] Muszyński W., Banaszak Z., Tomczuk-Piróg I.: *Automated vehicles' work planning in flexible manufacturing systems*, Proc. of the 11th IEEE Int. Conf. on Emerging Technologies and Factory Automation, Prague, Czech Republic, 2006, pp.813-818.

[61] Niederliński A.: *Constraint Logic Programming – From Prolog to CHIP*, Proceedings of the Workshop on Constraint Programming for Decision and Control, Gliwice, 1999, pp. 27-34.

[62] Niederliński A.: *Regułowe systemy ekspertowe*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice, 2000.

[63] Orski D., *Analiza i podejmowanie decyzji w systemie ekspertowym z hybrydową reprezentacją wiedzy*, praca doktorska, Politechnika Wrocławska, 2000.

[64] Pareschi R.: *Work force management in telecommunications and beyond telecommunications.* In Proc. PACLP99, 1999.

[65] Piegat A.: *Modelowanie i sterowanie rozmyte*, Exit, 1998.

[66] Pires J.M., Dubois D., Prade H.: *Fuzzy Constraint Problems with General Aggregation Operations under Possibilistic Logic Form*, The European Congress on Intelligent Techniques and Soft Computing, EUFIT '98, 1998, pp. 535 – 540.

[67] Perrett M.: *Using constraint logic programming techniques in container port planning.* ICL Technical Journal, 1991, pp. 537 – 545.

[68] Polak M., Majdzik P., Banaszak Z.A., Wójcik R.: The performance evaluation tool for automated prototyping of concurrent cyclic processes. *Fundamenta Informaticae*, Vol.60, No.1–4, 2004, pp. 269 – 289.

[69] Puget J-F.: *A C++ Implementations of CLP*, Proceeding of SPICS 94, 1994.

[70] Ramamritham K.: *Allocation and scheduling of precedence-related periodic tasks*, IEEE Trans. On Parallel and Distributed Systems, No 6, Vol.4, 1995, pp. 412 – 420.

[71] Rossi F.: *Constraint (Logic) programming: A Survey on Research and Applications*, K.R. Apt et al. (Eds.), New Trends in Constraints, LNAI 1865, Springer-Verlag, Berlin, 2000, pp. 40 – 74.

[72] Saniuk S., Witkowski K., Mądry M.: *Metoda planowania przepływu produkcji w warunkach deterministycznych ograniczeń logistycznych.* V Konferencja Naukowa Nowoczesne Zarządzanie Przedsiębiorstwem, Zielona Góra - Dychów, 2000, str. 332 – 342.

[73] Saniuk S., Woźniak W., Mądry M.: *Planowanie przepływu wieloasortymentowej produkcji rytmicznej w warunkach ograniczeń logistycznych.* IV Krajowa Konferencja Naukowa, Nowoczesne Zarządzanie Przedsiębiorstwem, Zielona Góra, 1999, str. 414 – 424.

[74] Schrage, L.: Cunningham K.: *Language for INteractive General Optimization*, version Demo LINGO/PC 1.04a, LINDO Systems Inc., Chicago, 1988.

[75] Schulte CH.: *Programming Constraint Services*, praca doktorska, Saarbrucken, 2000.

[76] Skołud B. (red.): *Systemy wspomagania decyzji w planowaniu produkcji*, Monografia, Praca zbiorowa, Wydawnictwo Politechniki Śląskiej, Gliwice, 2001.

[77] Schulte CH., Smolka G., Wurtz J.: *Finite Domain Constraint Progarmming in Oz,* DFKI OZ documentaion series, German Research Center for Artificial Inteligence, Stuhlsaltzenhausweg 3, D-66123 Saarbrucken, Geramny, 1998.

[78] Simonis H.: *Trends In Finite Domain Constraint Programming,* Proceedings of the Workshop on Constraint Programming for decision and Control, Gliwice, 1999.

[79] Sitek P., Wikarek J.: *System wspomagania harmonogramowania produkcji z wykorzystaniem metodyki programowania w logice z ograniczeniami*, Metody i techniki zarządzania w inżynierii produkcji, Bielsko-Biała, 2006, str.161 – 181.

[80] Sitek P., Wikarek J., Banaszak Z.: *Zastosowanie metodyki programowania w logice z ograniczeniami do optymalizacji planowania zleceń produkcyjnych*, Red. Zaborowski M., Automatyzacja Procesów Dyskretnych, WNT, Warszawa, 2004, s. 295 – 304.

[81] Tomczuk I.: *Zastosowanie technik programowania CLP do planowania przepływu produkcji w małych i średnich przedsiębiorstwach przemysłu maszynowego*, praca doktorska, Politechnika Warszawska, 2005.

[82] Tomczuk I., Banaszak Z., Jakubowski J., Bzdyra K.: *Planowanie przewozów w miejskich sieciach dystrybucji towarów*, Logistyka a infrastruktura miejska, Wrocław 2004, str. 150 – 157.

[83] Tomczuk-Piróg I., Muszyński W., **Bocewicz G.**: *CLP – Based Approach to Decision Making Aimed at Production Orders Prototyping*, 12th IEEE International Conference on Methods and Models in Automation and Robotics, Międzyzdroje 2006, pp.1091 – 1096.

[84] Topolska S.: *Process of vacuum metallization – a simulation in the taylor program*, Journal of Achievements in Materials and Manufacturing Engineering, vol. 16 ISSUE 1-2, 2006.

[85] Wallace M.: *Practical applications of constraint programming*, Constraints Journal, Vol. 1(1), 1996, pp. 139 – 168.

[86] Wallace M.: *Constraint Logic Programming*, Ed. Kakas A.C., Sadri F., Computat. Logic, LNAI 2407, Springer-Verlag, Berlin, Heidelberg, 2000, pp. 512 – 532.

[87] Wójcik R.: *Constraint Programming Approach to Design Conflict-Free Schedules for Repetitive Manufacturing Systems*, Proc. of the 3rd CIRP Conference on Digital Enterprise Technology, P.F. Cunha, P.G. Maropoulos (Eds.), Setubal, Portugal, 2006, P6.

[88] Wójcik R., Banaszak Z., Józefczyk J.: *CP approach to design of concurrently competing process flows in repetitive manufacturing systems*, Proc. of the 11th IEEE International Conference on Methods and Models in Automation and Robotics, Międzyzdroje, 2005, pp. 861 – 866.

[89] Wójcik R., **Bocewicz G.**: *CP-approach to design of steady-state flow of repetitive manufacturing processes in SME,* Applied computer science and production management. Eds: Z. Banaszak, J. Matuszek,. Apllied Computer Science Vol.1, No 1, 2005, Wydawnictwo ATH w Bielsku Białej, 2005, pp 201 – 218.

[90] Wójcik R., **Bocewicz G.**, Banaszak Z.: *CP approach to prototyping of systems of concurrently competing cyclic processes using a logic-algebraic method*, The Eighth International Conference on Artificial Intelligence and Soft Computing, ICAISC, Zakopane, 2006, pp 308 – 315.

[91] Wójcik R., **Bocewicz G.**, Banaszak Z.: *Knowledge engineering approach to concurrently competing cyclic processes control* The 3rd IEEE Conference On Intelligent Systems, IEEE IS'06, London, 2006, pp. 77 – 82.

[92] Wójcik R., **Bocewicz G.**, Banaszak Z.: *Harmonogramowanie pracy wózków samojezdnych w warunkach ograniczonego dostępu do współdzielonych zasobów ESW*

(Model logiczno-algebraiczny), IX Konferencja Automatyki Robotyki, KKR, Piechowice, 2006, str. 149 – 163.

[93] Wójcik R., **Bocewicz G.**, Banaszak Z.: *Zastosowania technik programowania z ograniczeniami do rozstrzygania konfliktów zasobowych w ESP*, Efektywność obrabiarek i procesów produkcyjnych, Inżynieria Maszyn Zeszyt 1, 2006, str. 87 – 99.

[94] Van Roy P., Haridi S.: *Programowanie Koncepcje techniki i modele*, Helion 2005.

[95] Zawadzka L.: *Podstawy projektowania elastycznych systemów sterowania produkcją*, Wyd. Politechniki Gdańskiej, Gdańsk, 2000.

[96] Ziembiński Z.: *Logika Praktyczna*, PWN, Warszawa, 2000,

[97] http://www.en.wikipedia.org

[98] http://www.at-consulting.de/ilogCLP/optitrans.htm

[99] http://www.ilog.com

[100] http://www.lindo.com

[101] http://www.ats4com.wedb.pl

[102] http://www.taylor.com

[103] http://www.swi-prolog.org

Dodatek A. Wyniki eksperymentów

Badania *SSD* przeprowadzone zostały na komputerze z procesorem Duo T2300E i pamięcią RAM: 512 MB. Eksperymenty zrealizowane zostały dla systemów o współczynnikach nasycenia 0,5; 0,3; 0,4; 0,7 (do grupy o współczynniku nasycenia 0,5 zaliczane były systemy o współczynniku $4{,}75 \leq G_T \leq 5{,}25$ – analogicznie przydział odbywał się dla pozostałych grup) i liczby procesów z zakresu 8 – 25.

W tabelach Tab. A.1, Tab. A.2 zestawione zostały wyniki eksperymentów dla $G_T = 0{,}5$. Eksperymenty obejmowały pomiar czasu kompilacji (traktowany jako czas generowania reprezentacji wiedzy oraz czas kompilacji plików wsadowych środowiska Oz Mozart), czasu wyznaczenia pierwszego warunku wystarczającego, czasu wyznaczenia wszystkich warunków wystarczających oraz pomiaru liczby wyznaczonych warunków. W prezentowanych tabelach czas jest wyrażony w sekundach. Każdy wiersz tabeli odpowiada przeprowadzonej serii pomiarów. Tabele zawierają średnie wartości uzyskanych wyników. Dla liczby procesów od 8 do 14 w serii realizowanych było po 10 pomiarów (tzn. dla 10 systemów transportowych o tych samych parametrach), od 14 do 19 po 4 pomiary, dla liczby procesów powyżej 19 po 2 pomiary. Tabela A.1 ilustruje wyniki pomiarów przy braku dodatkowych ograniczeń (poszukiwane są rozwiązania bezblokadowe i bezkolizyjne).

Tab. A.1. Czasy wyznaczania warunków wystarczających (przy braku ograniczeń) dla systemów o współczynniku nasycenia $G_T = 0{,}5$.

$G_T = 0{,}5$	Brak ograniczeń				
		Czasy wyznaczania warunków wystarczających [s]			
Liczba procesów	Liczba zasobów	Czas kompilacji	Czas wyznaczania pierwszego warunku	Czas rozwiązania	Liczba warunków
*8	14	20,515	3,432	57,734	318
9	15	40,543	0,757	55,056	247
10	21	71,123	0,709	73,047	186
11	21	90,719	24,675	95,547	10704
12	21	85,432	2,432	122,898	7528
13	21	102,131	15,184	197,234	1145
14	21	133,625	1,029	250,375	14060
15	22	151,432	0,769	503,597	24584
16	23	234,543	2,761	800,778	35871
17	23	447,76	7,067	1662,670	90657
18	25	301,431	4,560	3304,091	67524
19	27	454,788	10,112	5546,949	87414
20	30	480,432	12,463	10524,303	94545
21	32	543,593	9,493	19151,853	114521
23	36	756,435	0,709	42050,452	188547
25	38	868,017	15,133	72014,908	425158

* - przykłady takich systemów zostały przedstawione na końcu dodatku A.

Tabela A.2 zawiera wyniki pomiarów dla tych samych systemów transportowych przy dodatkowym ograniczeniu czasu trwania cyklu. Założono, że cykl nie może być większy niż $1{,}3 \cdot T_M$, gdzie T_M - suma czasów trwania operacji najdłuższej marszruty. Poszukiwane były warunki gwarantując brak kolizji, blokady oraz spełniające ograniczenie długości cyklu.

Tab. A.2. Czasy wyznaczania warunków wystarczających dla systemów o współczynniku nasycenia $G_T = 0,5$ przy dodatkowym ograniczeniu czasu trwania cyklu.

$G_T = 0,5$	Dodatkowe ograniczenia na czas trwania cyklu				
		Czasy wyznaczania warunków wystarczających [s]			
Liczba procesów	Liczba zasobów	Czas kompilacji	Czas wyznaczania pierwszego warunku	Czas rozwiązania	Liczba warunków
8	14	21,114	0,765	2,112	23
9	15	39,096	0,708	9,023	31
10	21	71,564	0,721	7,021	28
11	21	90,645	0,742	5,449	78
12	21	85,432	1,321	8,212	55
13	21	102,285	0,804	15,842	43
14	21	133,098	2,342	23,544	234
15	22	151,432	0,769	62,321	121
16	23	235,198	2,761	92,021	323
17	23	447,553	9,543	154,432	564
18	25	300,727	16,865	432,654	213
19	27	454,788	10,654	723,032	565
20	30	481,357	44,565	3476,786	474
21	32	543,986	32,543	3943,756	1232
23	36	758,211	0,743	5431,001	2312
25	38	867,346	65,453	11231,121	6431

Tabele A.3, A.4, A.5 zawierają zestawienia wyników eksperymentów dla systemów o współczynnikach gęstości 0,3; 0,4; 0,7. Eksperymenty przeprowadzone zostały przy braku dodatkowych ograniczeń.

Tab. A.3. Czasy wyznaczania warunków wystarczających (przy braku ograniczeń) dla systemów o współczynniku nasycenia $G_T = 0,7$.

$G_T = 0,7$	Brak ograniczeń				
		Czasy wyznaczania warunków wystarczających [s]			
Liczba procesów	Liczba zasobów	Czas kompilacji	Czas wyznaczania pierwszego warunku	Czas rozwiązania	Liczba warunków
8	20	22,534	0,719	241,156	29616
9	21	25,937	0,757	405,554	120432
10	25	56,593	0,701	552,908	60341
*11	25	68,78	0,786	678,656	245321
12	25	74,412	0,702	1054,221	190321
13	25	71,121	0,732	2140,009	195322
14	25	86,037	1,231	3220,321	285662
15	26	211,431	1,369	4548,012	403219
16	28	234,783	0,726	5564,653	231021
17	28	313,212	0,702	12541,902	599502
18	30	292,101	1,432	18547,776	805321
19	32	308,554	0,801	32140,321	1190594
20	35	317,432	0,943	62351,374	1388741

* - przykłady takich systemów zostały przedstawione na końcu dodatku A.

Tab. A.4. Czasy wyznaczania warunków wystarczających (przy braku ograniczeń) dla systemów o współczynniku nasycenia $G_T = 0,3$.

$G_T = 0,3$	Brak ograniczeń				
		Czasy wyznaczania warunków wystarczających [s]			
Liczba procesów	Liczba zasobów	Czas kompilacji	Czas wyznaczania pierwszego warunku	Czas rozwiązania	Liczba warunków
8	14	208,531	5,789	149,834	54
9	15	223,421	12,154	210,064	81
10	21	211,432	23,435	310,656	43
11	21	270,433	35,458	409,912	60
12	21	332,874	21,903	423,876	80
13	21	254,332	13,732	457,806	154
14	21	434,322	60,231	796,582	45
15	22	318,625	134,369	938,375	143
16	23	360,221	321,726	2102,178	21
17	23	398,765	215,702	3832,890	56
18	25	450,543	546,432	7201,017	613
19	27	503,981	321,801	11431,287	3254
*20	27	617,764	654,943	13629,491	2176
21	32	634,532	1212,976	32283,679	471
23	36	764,342	1056,016	48758,136	1204
25	38	723,006	678,434	76027,773	5247

* - przykłady takich systemów zostały przedstawione na końcu dodatku A.

Tab. A.5. Czasy wyznaczania warunków wystarczających (przy braku ograniczeń) dla systemów o współczynniku nasycenia $G_T = 0,4$.

$G_T = 0,4$	Brak ograniczeń				
		Czasy wyznaczania warunków wystarczających [s]			
Liczba procesów	Liczba zasobów	Czas kompilacji	Czas wyznaczania pierwszego warunku	Czas rozwiązania	Liczba warunków
8	14	220,312	4,789	65,5	54
9	15	242,055	6,757	87	81
10	21	213,133	0,719	120,469	43
11	21	332,095	0,707	189	60
12	21	355,765	2,802	213	80
13	21	254,332	3,732	243	154
*14	21	434,322	0,743	512	45
15	22	443,122	5,369	767	143
16	23	462,084	19,054	1232	21
17	23	431,431	23,432	2134	56
18	25	410,131	26,325	3843	613
19	27	523,232	65,801	8675	3254
20	30	654,767	87,943	15321	2176
21	32	623,387	134,432	25236	471
22	36	864,345	345,801	40257	1204
23	38	897,654	4,789	57268	5247

* - przykłady takich systemów zostały przedstawione na końcu dodatku A.

Tabele A.6, A.7, zawierają wyniki eksperymentów polegających na pomiarze czasu generowania gotowego już harmonogramu w oparciu o wyznaczone wcześniej warunki wystarczające (tabele A.1 do A.5). Podobnie jak w przypadku poprzednich eksperymentów tabele zawierają wartości średnie otrzymanych wyników.

Tab. A.6. Czasy wyznaczania harmonogramów w oparciu o uprzednio wyznaczone warunki wystarczające dla systemów o współczynniku nasycenia $G_T = 0,3$, $G_T = 0,4$.

	Czasy dla $G_T = 0,3$ [s]			Czasy dla $G_T = 0,4$ [s]		
Procesy	kompilacji	rozwiązania	suma	kompilacji	rozwiązania	suma
8	4,034	0,76	4,794	4,434	0,76	5,194
9	4,321	0,723	5,044	4,121	1,201	5,322
10	4,322	0,745	5,067	4,887	0,745	5,632
11	4,875	1,123	5,998	4,235	0,725	4,96
12	4,221	0,743	4,964	4,565	0,713	5,278
13	4,765	0,781	5,546	4,211	0,7921	5,0031
14	4,871	1,32	6,191	4,001	1,32	5,321
15	4,761	0,751	5,512	4,121	0,721	4,842
16	4,321	0,742	5,063	4,982	0,742	5,724
17	4,322	0,746	5,068	5,212	0,746	5,958
18	4,275	0,764	5,039	4,761	0,814	5,575
19	4,281	0,765	5,046	4,112	0,765	4,877
20	4,965	0,821	5,786	4,09	0,721	4,811
21	4,076	1,22	5,296	4,231	0,722	4,953
23	4,761	0,761	5,522	4,221	0,861	5,082
25	4,982	0,742	5,724	4,001	0,713	4,714

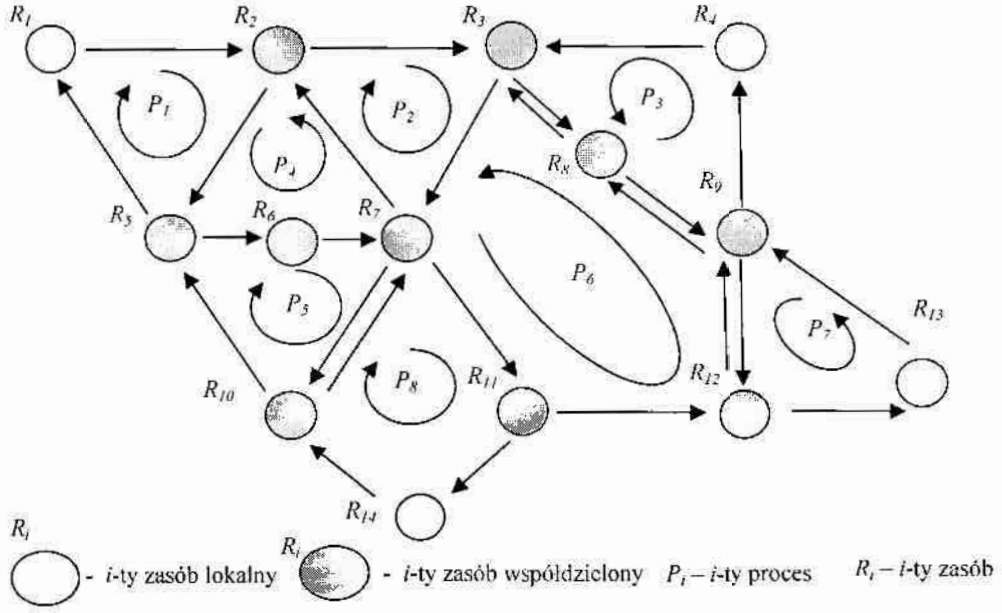
Tab. A.7. Czasy wyznaczania harmonogramów w oparciu o uprzednio wyznaczone warunki wystarczające dla systemów o współczynniku nasycenia $G_T = 0,5$, $G_T = 0,7$.

	Czasy dla $G_T = 0,5$ [s]			Czasy dla $G_T = 0,7$ [s]		
Procesy	kompilacji	rozwiązania	suma	kompilacji	rozwiązania	suma
8	4,134	0,76	4,894	4,114	0,76	4,874
9	4,221	0,701	4,922	4,121	0,701	4,822
10	4,687	0,745	5,432	4,211	0,715	4,926
11	4,235	0,723	4,958	4,001	0,713	4,714
12	4,034	0,787	4,821	4,121	0,717	4,838
13	4,321	0,781	5,102	4,182	0,721	4,903
14	4,322	0,732	5,054	4,212	0,732	4,944
15	4,875	0,711	5,586	4,275	0,711	4,986
16	4,494	0,762	5,256	4,494	0,722	5,216
17	4,100	0,743	4,843	4,100	0,713	4,813
18	4,522	0,781	5,303	4,322	0,741	5,063
19	4,414	0,82	5,234	4,014	0,72	4,734
20	4,121	0,751	4,872	4,121	0,751	4,872
21	4,887	0,792	5,679	4,187	0,712	4,899
22	-	-	-	4,135	0,731	4,866
23	4,235	0,731	4,966	4,114	0,76	4,874
25	4,076	1,22	5,296	-	-	-

Przykładowe struktury SWPC

Na rysunkach A.1, A.2, A.3, A.4, przedstawione zostały wybrane struktury *SWPC* reprezentujące systemy transportowe, dla których przeprowadzone zostały pomiary.

Struktura 1: 8 procesów, 14 zasobów – system z rysunku A.1 odpowiada systemowi transportowemu z tabeli A.1, dla 8 procesów.



Rys. A.1. Struktura SWPC odpowiadająca systemowi z 14 zasobami i 8 procesami.

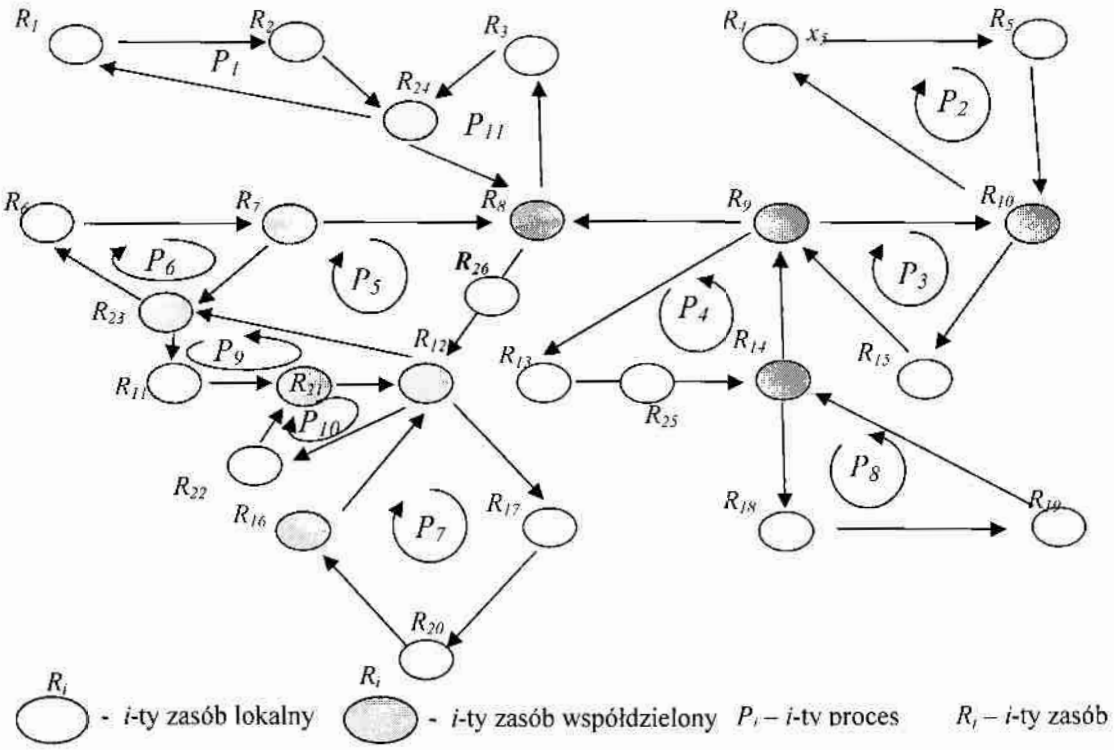
W systemie procesy realizują następujące marszruty:

$P_1 = (R_1, R_2, R_5)$, $P_2 = (R_2, R_3, R_7)$, $P_3 = (R_3, R_4, R_9, R_8)$, $P_4 = (R_2, R_5, R_6, R_7)$, $P_5 = (R_5, R_6, R_7, R_{10})$, $P_6 = (R_3, R_7, R_{11}, R_{12}, R_9, R_8)$, $P_7 = (R_9, R_{12}, R_{13})$, $P_8 = (R_7, R_{11}, R_{14}, R_{10})$.

Czasy trwania operacji elementarnych są następujące:

$T_1 = (1, 2, 3)$, $T_2 = (1, 2, 3)$, $T_3 = (1, 2, 3, 4)$, $T_4 = (1, 2, 3, 4)$, $T_5 = (1, 2, 3, 4)$, $T_6 = (1, 1, 2, 2, 3, 3)$, $T_7 = (1, 2, 3)$, $T_8 = (1, 2, 3, 4)$.

Struktura 2: 11 procesów, 25 zasobów – system z rysunku A.2 odpowiada systemowi transportowemu z tabeli A.3, wiersz: 11 procesów.



Rys. A.2. Struktura SWPC odpowiadająca systemowi z 25 zasobami i 11 procesami.

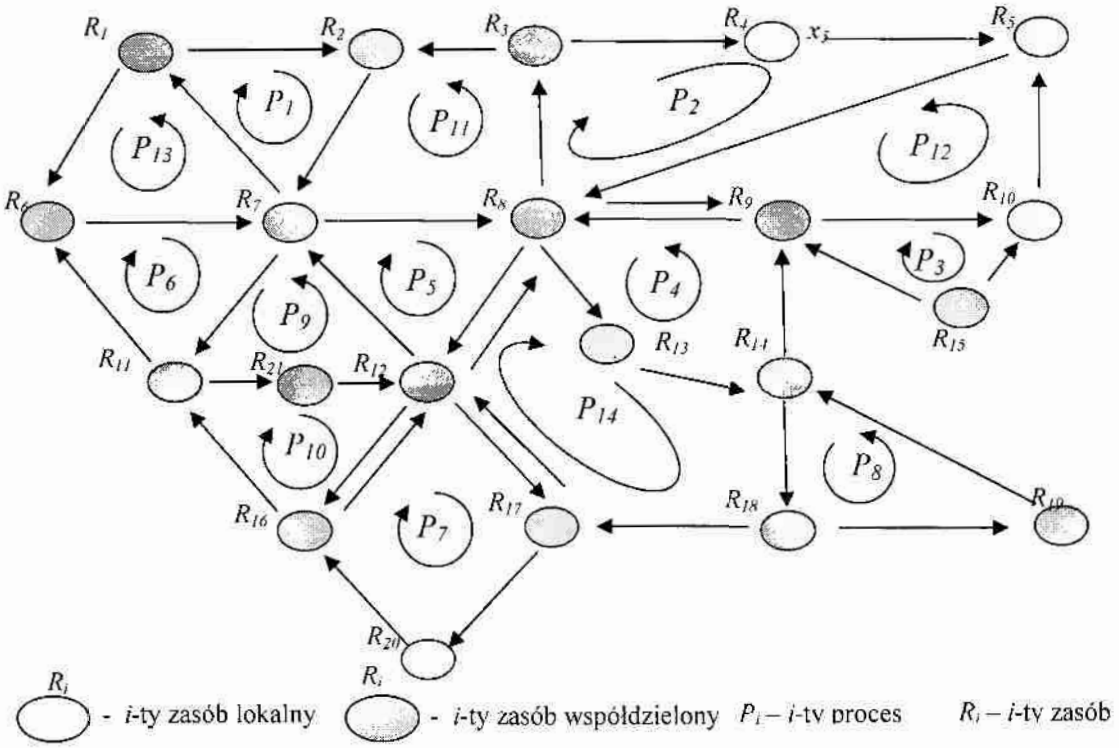
W systemie procesy realizują następujące marszruty:

$P_1 = (R_1, R_2, R_{24})$, $P_2 = (R_4, R_5, R_{10})$, $P_3 = (R_9, R_{10}, R_{15})$, $P_4 = (R_9, R_{13}, R_{25}, R_{14})$,
$P_5 = (R_7, R_8, R_{26}, R_{12}, R_{23})$, $P_6 = (R_6, R_7, R_{23})$, $P_7 = (R_{16}, R_{12}, R_{17}, R_{20})$, $P_8 = (R_{14}, R_{18}, R_{19})$, $P_9 = (R_{12}, R_{23}, R_{11}, R_{21})$, $P_{10} = (R_{21}, R_{12}, R_{11})$.

Czasy trwania operacji elementarnych są następujące:

$T_1 = (1, 2, 3)$, $T_2 = (1, 2, 3)$, $T_3 = (1, 2, 3)$, $T_4 = (1, 2, 3, 4)$, $T_5 = (1, 2, 3, 4, 1)$,
$T_6 = (1, 1, 2)$, $T_7 = (1, 2, 3, 1)$, $T_8 = (1, 2, 3)$, $T_9 = (1, 2, 3, 1)$, $T_{10} = (1, 2, 3)$.

Struktura 3: 14 procesów, 21 zasobów – system z rysunku **A.3** odpowiada systemowi transportowemu z tabeli A.5, dla 14 procesów.



Rys. A.3. Struktura SWPC odpowiadająca systemowi z 21 sektorami i 14 procesami.

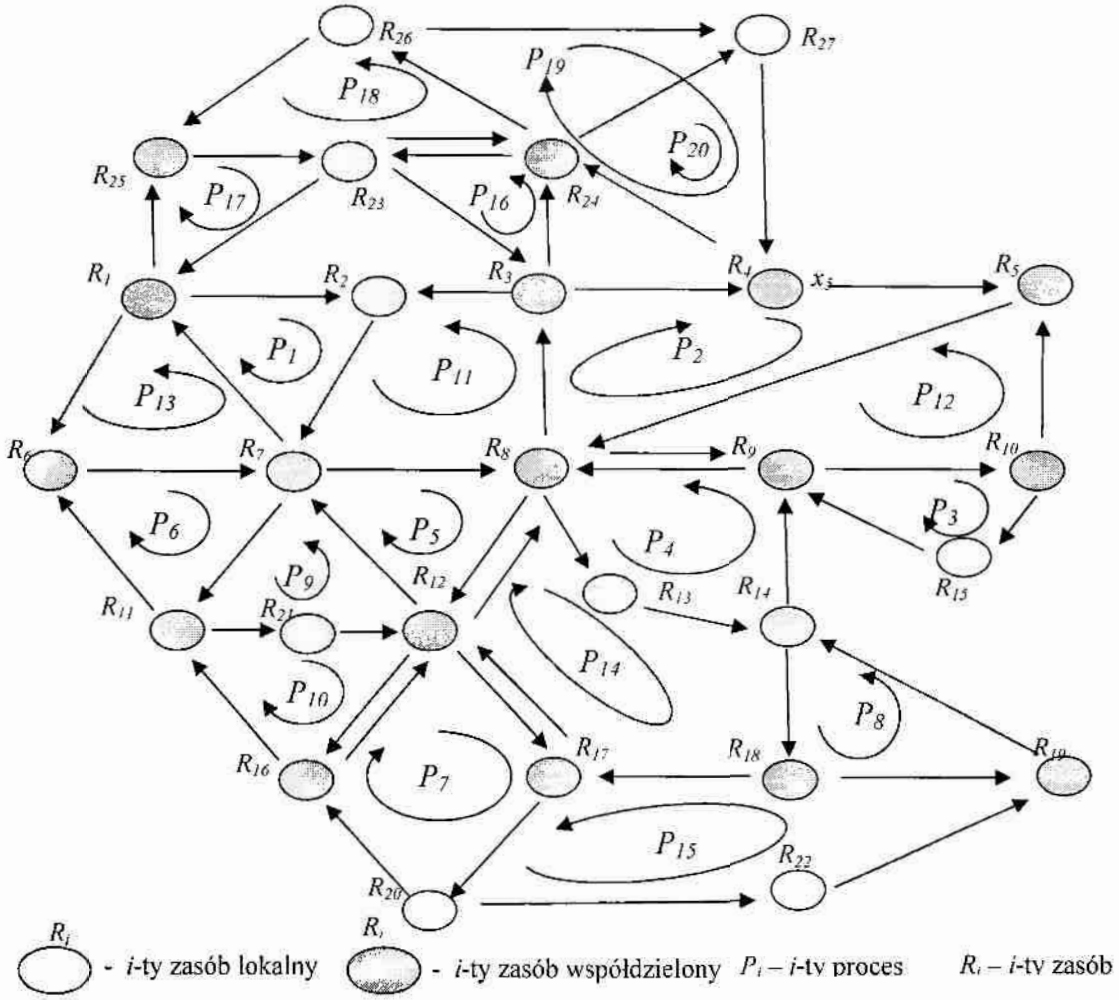
W systemie procesy realizują następujące marszruty:

$P_1 = (R_1, R_2, R_7)$, $P_2 = (R_4, R_5, R_8, R_3)$, $P_3 = (R_9, R_{10}, R_{15})$, $P_4 = (R_9, R_8, R_{13}, R_{14})$,
$P_5 = (R_7, R_8, R_{12})$, $P_6 = (R_6, R_7, R_{11})$, $P_7 = (R_{16}, R_{12}, R_{17}, R_{20})$, $P_8 = (R_{14}, R_{18}, R_{19})$,
$P_9 = (R_{12}, R_7, R_{11}, R_{21})$, $P_{10} = (R_{21}, R_{12}, R_{16}, R_{11})$, $P_{11} = (R_2, R_7, R_8, R_3)$,
$P_{12} = (R_5, R_8, R_9, R_{10})$, $P_{13} = (R_1, R_6, R_7)$, $P_{14} = (R_8, R_{13}, R_{14}, R_{18}, R_{17}, R_{12},)$.

Czasy trwania operacji elementarnych są następujące:

$T_1 = (1, 2, 3)$, $T_2 = (1, 2, 3, 1)$, $T_3 = (1, 2, 3)$, $T_4 = (1, 2, 3, 4)$, $T_5 = (1, 2, 3,)$,
$T_6 = (1, 1, 2)$, $T_7 = (1, 2, 3, 1)$, $T_8 = (1, 2, 3)$, $T_9 = (1, 2, 3, 1)$, $T_{10} = (1, 2, 3, 4)$,
$T_{11} = (1, 2, 3, 4)$, $T_{12} = (1, 2, 3, 4)$, $T_{13} = (1, 2, 3)$, $T_{14} = (1, 2, 3, 1, 2, 3)$,

Struktura 4: 20 procesów, 27 zasobów – system z rysunku A.4 odpowiada systemowi transportowemu z tabeli A.4, dla 20 procesów.



Rys. A.4. Struktura SWPC odpowiadająca systemowi z 14 sektorami i 8 procesami.

W systemie procesy realizują następujące marszruty:

$P_1 = (R_1, R_2, R_7)$, $P_2 = (R_4, R_5, R_8, R_3)$, $P_3 = (R_9, R_{10}, R_{15})$, $P_4 = (R_9, R_8, R_{13}, R_{14})$, $P_5 = (R_7, R_8, R_{12})$, $P_6 = (R_6, R_7, R_{11})$, $P_7 = (R_{16}, R_{12}, R_{17}, R_{20})$, $P_8 = (R_{14}, R_{18}, R_{19})$, $P_9 = (R_{12}, R_7, R_{11}, R_{21})$, $P_{10} = (R_{21}, R_{12}, R_{16}, R_{11})$, $P_{11} = (R_2, R_7, R_8, R_3)$, $P_{12} = (R_5, R_8, R_9, R_{10})$, $P_{13} = (R_1, R_6, R_7)$, $P_{14} = (R_8, R_{13}, R_{14}, R_{18}, R_{17}, R_{12},)$, $P_{15} = (R_{17}, R_{20}, R_{22}, R_{19}, R_{18})$, $P_{16} = (R_{23}, R_3, R_{24})$, $P_{17} = (R_{25}, R_{23}, R_1)$, $P_{18} = (R_{23}, R_{24}, R_{26}, R_{25})$, $P_{19} = (R_{26}, R_{24}, R_4, R_{27})$, $P_{20} = (R_{24}, R_4, R_{27})$.

Czasy trwania operacji elementarnych są następujące:

$T_1 = (1, 2, 3)$, $T_2 = (1, 2, 3, 1)$, $T_3 = (1, 2, 3)$, $T_4 = (1, 2, 3, 4)$, $T_5 = (1, 2, 3,)$, $T_6 = (1, 1, 2)$, $T_7 = (1, 2, 3, 1)$, $T_8 = (1, 2, 3)$, $T_9 = (1, 2, 3, 1)$, $T_{10} = (1, 2, 3, 4)$, $T_{11} = (1, 2, 3, 4)$, $T_{12} = (1, 2, 3, 4)$, $T_{13} = (1, 2, 3)$, $T_{14} = (1, 2, 3, 1, 2, 3)$, $T_{14} = (1, 2, 3, 1, 2)$, $T_{16} = (1, 2, 3)$, $T_{17} = (1, 2, 3)$, $T_{18} = (1, 2, 3, 1)$, $T_{19} = (1, 2, 3, 1)$, $T_{20} = (1, 2, 3)$.

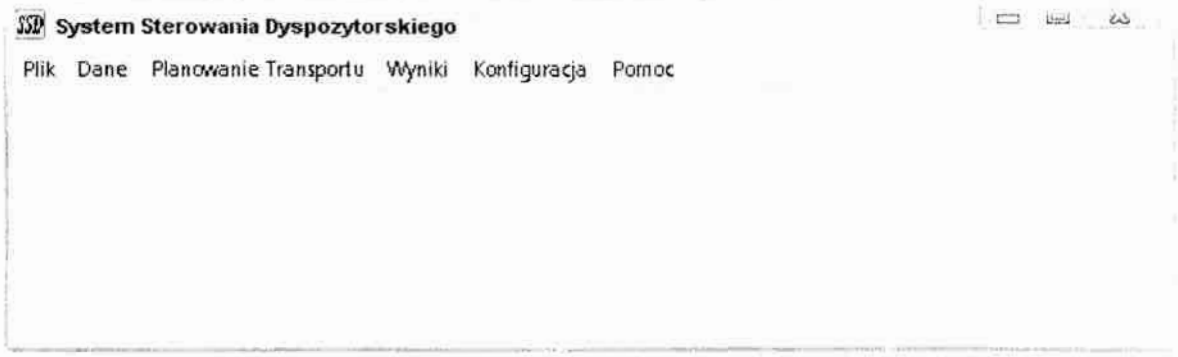
Dodatek B. System Sterowania Dyspozytorskiego – opis użytkowy

Niniejszy rozdział zawiera opis użytkowy, zbudowanego w ramach pracy, komputerowego *Systemu Sterowania Dyspozytorskiego* (*SSD*). System ten pozwala na wyznaczenie, w trybie interakcyjnym, dopuszczalnego harmonogramu pracy wózków samojezdnych, spełniającego ograniczenia zadane przez użytkownika systemu. Wyznaczenie harmonogramu polega na znalezieniu takiego planu realizacji poszczególnych operacji realizowanych procesów, który spełni ograniczenia wynikające ze struktury sytemu oraz ograniczenia użytkownika.

Połączenie zastosowanych środowisk informatycznych pozwoliło na opracowanie przyjaznego dla użytkownika programu, wykorzystującego zaawansowane techniki *CP*. Ukrycie warstwy obliczeń (język Oz Mozart), wykorzystującego zaawansowane techniki informatyczne za przyjaznym interfejsem użytkownika eliminuje konieczność zatrudniania wysokokwalifikowanej kadry programistów, którzy implementowaliby kolejne zadania w językach *CP*, a następnie analizowaliby uzyskane wyniki.

Aplikacja została przygotowana do pracy w systemach operacyjnych klasy Windows. Tym samym obsługa programu realizowana jest według standardów aplikacji „okienkowych". Zgodnie z tymi standardami, wywoływanie kolejnych funkcji (formularzy) programu realizowane jest poprzez wybranie ich kursorem myszy.

Główne okno aplikacji (rysunek B.2) zawiera kaskadowe menu, z którego uzyskuje się dostęp do poszczególnych grup funkcji programu.



Rys. B.2. Główne okno aplikacji

Funkcje pogrupowane są tematycznie. Można wyróżnić następujące bloki:

Plik: funkcje umożliwiające archiwizację ustawień i parametrów systemu (rysunek B.3). Odpowiadają one „klasycznym" funkcjom otwarcia i zapisu pliku. Pliki są zapisywane z rozszerzeniem *.boc.

Rys. B.3. Widok menu Plik

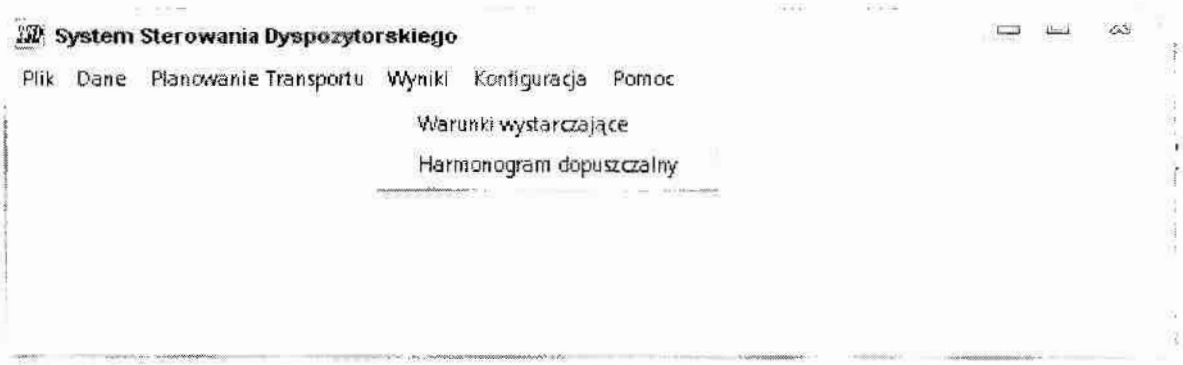
Dane: funkcje umożliwiające wprowadzenie zasobów produkcyjnych, procesów produkcyjnych, marszrut, czasów realizacji poszczególnych operacji oraz definiowania własnych ograniczeń (rysunek B.4);

Rys. B.4. Widok menu Dane

Planowanie Transportu: funkcje uruchomiania procedur wyznaczania warunków wystarczających oraz wyznaczania poszczególnych harmonogramów (rysunek B.5);

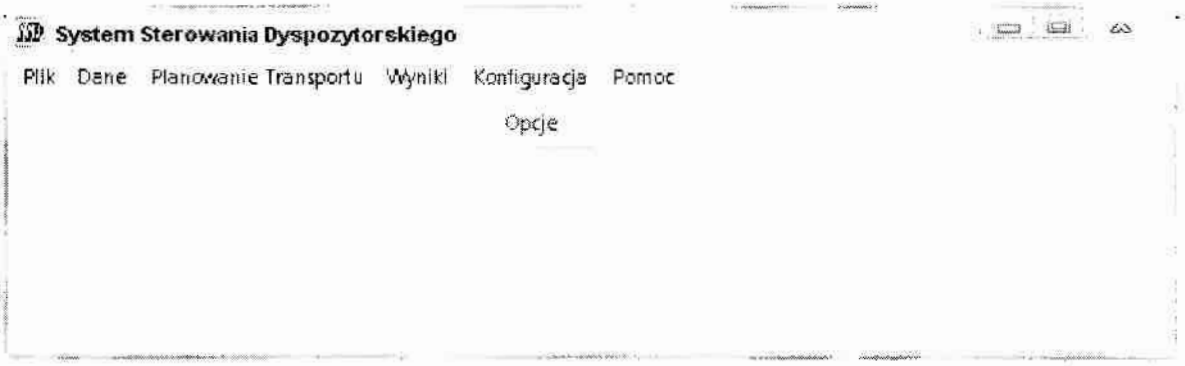
Rys. B.5. Widok menu Planowanie Transportu

Wyniki: funkcje umożliwiające prezentację warunków wystarczających, oraz gotowych harmonogramów (rysunek B.6);



Rys. B.6. Widok menu Wyniki

Konfiguracja: polecenie, które wywołuje formularz konfiguracji programu;



Rys. B.7. Widok menu Konfiguracja

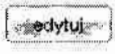
Pomoc: funkcje udzielające informacji o programie

W zbudowanym pakiecie do wprowadzania i edycji danych przeznaczono pasek nawigacji danych. Znajdują się na nim przyciski, które pozwalają na dopisywanie, edycję i usuwanie danych.

Znaczenie poszczególnych przycisków jest następujące:

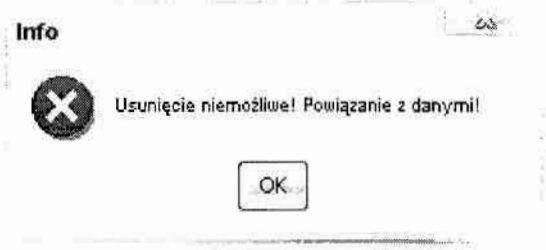

- dodawanie nowego rekordu po programu,


- usunięcie bieżącego rekordu z programu,



- edytowanie bieżącego rekordu.

W przypadku gdy usuwany rekord jest powiązany z innymi danymi, wówczas system informuje o tym użytkownika (rysunek B.8) a operacja usuwania rekordu jest anulowana.



Rys. B.8. Komunikat o powiązaniu usuwanych danych

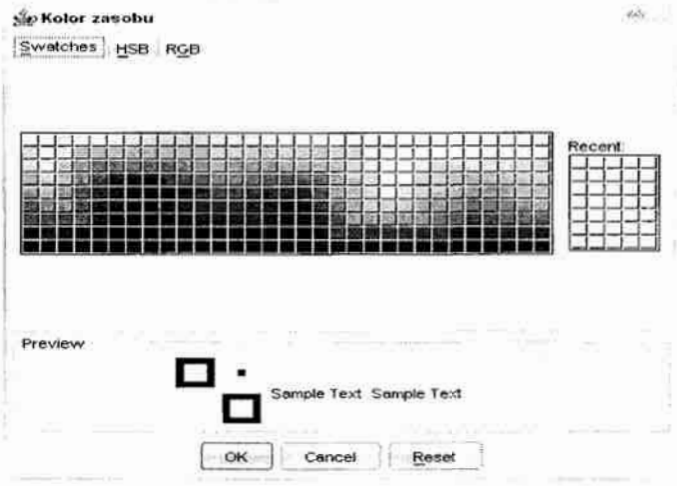
Wprowadzanie struktury sieci

W pierwszej kolejności wprowadzane do systemu są zasoby produkcyjne, ich nazwy oraz symbole, które będą wykorzystywane przy opisie harmonogramu. W tym celu z menu **Dane** należy wybrać funkcję **Zasoby**. Na ekranie pojawi się okno jak na rys. B.9.

Rys. B.9. Formularz wprowadzania zasobów produkcyjnych

W lewej części okna znajduje się lista wprowadzonych już do systemu typów zasobów, w części prawej pola edycyjne, gromadzonych danych. Znaczenie pól jest następujące:

- **Nazwa zasobu:** nazwa, która identyfikuje dany typ zasobu w systemie.
- **Symbol zasobu**: skrócony identyfikator typu zasobu. Symbol zasobu musi być unikalny, wykorzystywany on będzie do oznaczania zasobu na diagramów prezentacji wyników.
- **Kolor reprezentacji graficznej**: kolor, jakim dany typ zasobu będzie oznaczany na diagramach prezentacji wyników. W celu określenia koloru reprezentacji graficznej należy wcisnąć pole znajdujące się obok napisu *Kolor*. Otworzy się nowy formularz z paletą kolorów (rysunek B.10), w którym należy wybrać dowolny kolor i potwierdzić wybór klikając klawisz **OK**. Wciśnięcie klawisza **Anuluj** powoduje rezygnację z wyboru. Wybrany kolor wyświetli się w polu koloru.



Rys. B.10. Formularz wyboru koloru reprezentacji graficznej

Po wprowadzeniu zasobów należy w celu zapisania danych do programu wybrać przycisk **Zastosuj**.

Definiowanie procesów produkcyjnych

Po wprowadzeniu do systemu zasobów produkcyjnych należy wprowadzić parametry realizowanych procesów. W zbudowanym systemie składają się na nie marszruty, określające trasy poszczególnych procesów, oraz czasy realizacji poszczególnych operacji. Uwzględniając, że w przyjętym modelu marszruty mogą istnieć wyłącznie pomiędzy istniejącymi zasobami, konieczne jest wcześniejsze wprowadzenie do systemu zasobów produkcyjnych.

W celu wprowadzenia do systemu marszrut i czasów realizacji poszczególnych operacji, należy za pomocą funkcji **Procesy** (Menu **Dane**) otworzyć okno **Procesy produkcyjne**. Okno to (rysunek B.11) zawiera listę wprowadzonych już procesów oraz pola edycyjne umożliwiające ich edycję bądź wprowadzanie nowych procesów.

Rys. B.11. Okno Procesy Produkcyjne

Znaczenie kolumn, widocznych w liście wprowadzonych już do systemu procesów, jest następujące:

- **Nazwa**: nazwa procesu,
- **Symbol**: symboliczne oznaczenie procesu,
- **Marszruta**: lista zasobów tworzących marszrutę,
- **Czasy jednostkowe**: lista czasów trwania operacji na poszczególnych zasobach.

Znaczenie pól edycyjnych jest następujące:

- **Nazwa**: unikalna nazwa procesu,
- **Symbol**: symbol procesu (litera „P" i numer procesu),
- **Marszruta**: lista symboli zasobów, zawarta w nawiasach kwadratowych [],

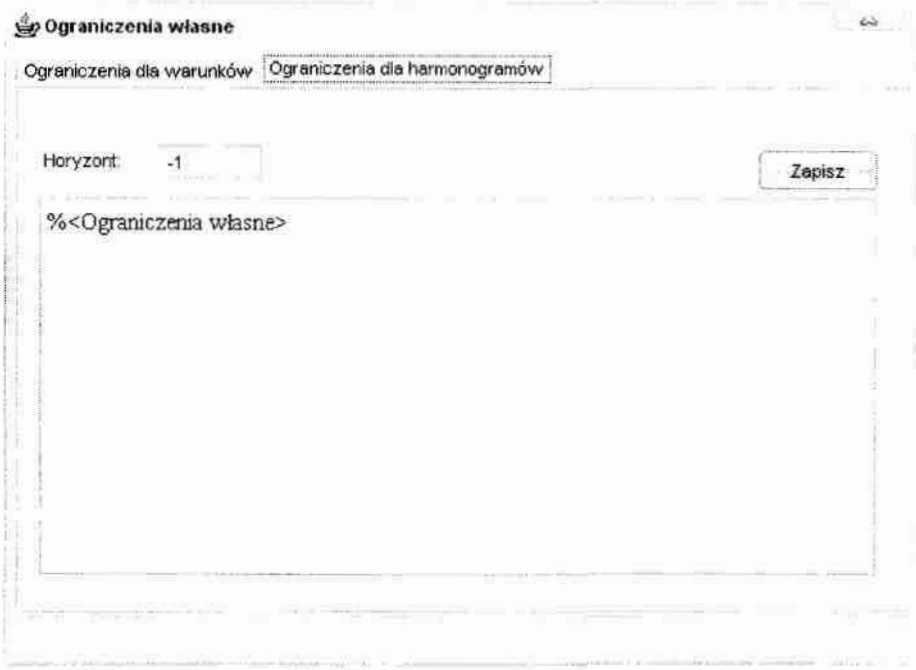
⇨ **Czasy jednostkowe** lista czasów trwania operacji na poszczególnych zasobach (zawarta w nawiasach kwadratowych []).

Pola **Marszruty** i **Czasy jednostkowe** są zbudowane w postaci list. W listach (koniecznie w nawiasach kwadratowych) wprowadza się symbole określające zasoby (marszruty) i czasy jednostkowe (wyrażone w umownych jednostkach czasu *ujc*) trwania operacji. Czas trwania operacji każdej z tras musi być liczbą naturalną większą od 0.

Dane wprowadzane są do systemu poprzez przycisk **Zastosuj**.

Definiowanie własnych ograniczeń

Jeśli istnieje taka potrzeba, użytkownik ma możliwość wprowadzenia własnych ograniczeń, które będą uwzględniane w procesie wyznaczania warunków wystarczających jak i w procesie wyznaczania harmonogramów.



Rys. B.12. Okno Ograniczenia własne

W celu wprowadzenia do systemu ograniczeń należy za pomocą funkcji **Ograniczenia własne** (menu **Dane**) otworzyć okno **Ograniczenia własne** (rysunek B.12). Okno zawiera dwie zakładki: *Ograniczenia dla warunków*, *Ograniczenia dla harmonogramów*. Układ każdej zakładki jest taki sam, zawierają one pole horyzontu, pole tekstowe do wprowadzania ograniczeń oraz przycisk **Zapisz**.

Pole *Horyzont* służy do określania długości trwania cyklu realizacji procesów. Domyślna wartość pola wynosi -1 oznacza to, że długość cyklu jest dowolna. Wprowadzenie innej wartości (wymagana jest zawsze wartość ze zbioru liczb naturalnych) powoduje uwzględnienie na etapie wyznaczania rozwiązań ograniczenia czasu trwania cyklu. Poszukiwane są rozwiązania, których cykle będą mniejsze bądź równe wartości zadanej w polu *Horyzont*.

Pole tekstowe służy do wprowadzania własnych ograniczeń (w postaci zdań logicznych). Domyślna wartość pola to %<Ograniczenia własne>. Znak % oznacza symbol komentarza, wyrażenia występujące po tym znaku są ignorowane przez system.

Wprowadzone ograniczenia mogą mieć postać zdań logicznych i algebraicznych. Struktura budowanych zdań jest następująca:

$$\{ opl \{ opl\ opa\ ... \}\ \{ opl\ ... \} \} =: b$$

gdzie: $\{opl\ ...\}$ – oznacza operacje logiczną,
opa – operacja algebraiczna,
b – zmienna określająca wartość logiczną zdania.

Realizowane operacje dzielą się na dwie grupy: operacje logiczne (*opl*), operacje algebraiczne (*opa*). Wśród operacji logicznych wyróżnić można operacje jednoargumentowe i dwuargumentowe. Poniżej przedstawiono możliwe postacie operacji:

$\{opl\ \{opl\ ...\}\ \{opl...\}\}$,
$\{opl\ opa\ opa\}$,
$\{opl\ \{opl\ ...\}\}$
$\{opl\ opa\}$.

gdzie: *opa* – oznacza operację algebraiczną.

Pierwsze dwie postacie odpowiadają operacjom dwuargumentowym, kolejne operacjom jednoargumentowym. Wyrażenie *opl* określa rodzaj realizowanej operacji (AND, OR, NOT, IMPL, EQUI), kolejne wyrażenia stanowią jej argumenty. Argumentem operacji *opl* może być operacja logiczna lub operacja algebraiczna *opa*.

Wśród operacji logicznych dwuargumentowych wyróżnia się operacje:

$\{AND\ a\ b\}$ – operacja koniunkcji $a \wedge b$,
$\{OR\ a\ b\}$ – operacja alternatywy $a \vee b$,
$\{IMPL\ a\ b\}$ – operacja implikacji $a \Rightarrow b$,
$\{EQUI\ a\ b\}$ – operacja równoważności $a \Leftrightarrow b$,

gdzie: a, b – argumenty operacji.

Wśród operacji logicznych jednoargumentowych wyróżnia się operację:

$\{NOT\ a\}$ – operacja negacji $\neg a$,

gdzie: a – argument operacji.

Operacje algebraiczne *opa* stanowią operacje zdefiniowane na zmiennych decyzyjnych *var*. Ogólna postać operacji jest następująca:

$$opa(var\ \sim A\ var\ ...\ \sim A\ var)$$

gdzie: *var* – oznacza zmienną decyzyjną ,

~*A* – oznacza operator algebraiczny (+, -, *,\,=, ≤, ≥).

Wykorzystywane operatory algebraiczne postaci +, -,*,\, =, ≤, ≥, są kolejno zapisywane za pomocą symboli: +, -,*,\, =: , =<: >=:

Zmienne decyzyjne *var* są to terminy rozpoczęcia realizowanych operacji oraz czasy ich trwania. Terminy rozpoczęcia operacji są wyrażane w postaci ogólnej:

Pi.Rj

gdzie: Pi – oznacza proces P_i,

Rj – oznacza zasób R_j.

Wyrażenie to oznacza termin rozpoczęcia operacji procesu P_i na zasobie R_j. Podobnie definiowane są czasy trwania operacji:

T.ij

gdzie: T.ij – oznacza czas t_{ij}.

Wyrażenie to oznacza czas trwania operacji procesu P_i, na zasobie R_j.

W oparciu o przedstawioną gramatykę można wprowadzać do systemu własne ograniczenia. Przykładowo zdanie:

{AND (P1.R7 >: P5.R7) (P7.R14 <: P3.R8)} =:1,

oznacza, że operacja procesu P_1 na zasobie R_7 musi rozpocząć się po operacji procesu P_5 na zasobie R_7 i operacja procesu P_7 na zasobie R_{14} musi rozpocząć się przed operacją procesu P_3 na zasobie R_8.

W szczególnym przypadku wprowadzone zdania mogą być tylko w postaci operacji algebraicznych, na przykład:

(P1.R7 >: P5.R7) =: 1

W przypadkach, gdy zdanie algebraiczne jest prawdzie to nie jest konieczne przypisywanie im wartości logicznej 1. Powyżej przedstawione zdanie może być również wprowadzone do sytemu w postaci:

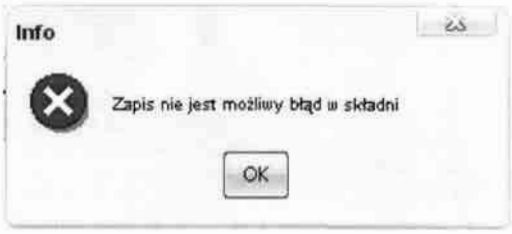
P1.R7 >: P5.R7

Podobnie zdanie {AND (P1.R7 >: P5.R7) (P7.R14 <: P3.R8)} =:1 może być przedstawione za pomocą dwóch zdań:

P1.R7 >: P5.R7
P7.R14 <: P3.R8.

Należy zaznaczyć, że w przypadku wprowadzania więcej niż jednego zdania, każde zdanie powinno być wpisywane w oddzielnym wierszu.

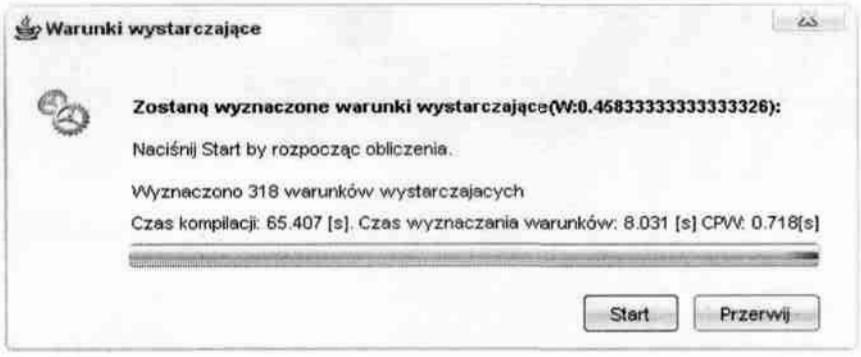
W celu uniknięcia wprowadzania błędnych zdań zastosowano kolorową składnię wprowadzanych symboli i poleceń.
W wprowadzenie danych do systemu odbywa się poprzez przycisk zapisz. W przypadku istnienia błędów w składni pojawia się komunikat o błędach (rysunek B.12).



Rys. B.13. Okno informacji o błędach.

Uruchamianie procedur przeszukiwania

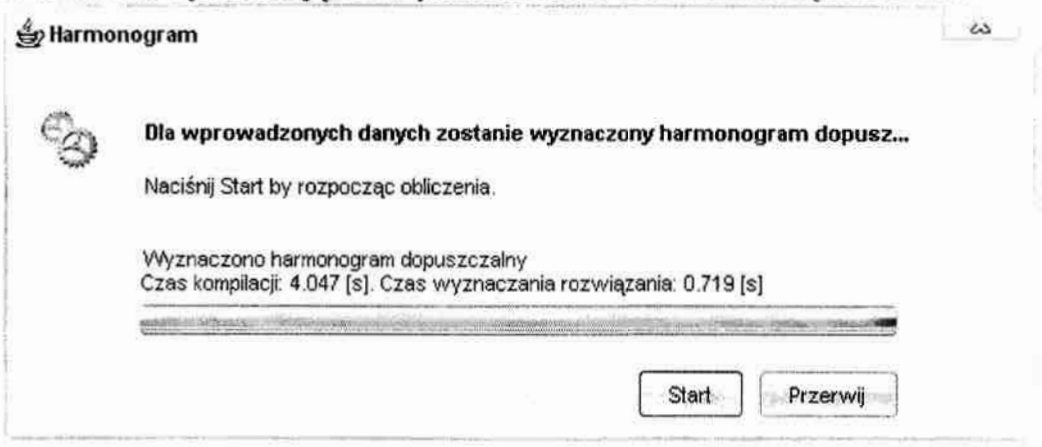
Po wprowadzeniu do systemu zasobów produkcyjnych, parametrów realizowanych procesów oraz dodatkowych ograniczeń, realizuje się proces wyznaczania warunków wystarczających. W celu uruchomienia procesu wyznaczania warunków wystarczających, należy za pomocą funkcji **Wyznacz warunki** (Menu **Planowanie Transportu**) otworzyć okno **Warunki wystarczające**. Okno to (rysunek B.13) zawiera przyciski **Start** i **Przerwij**. Przyciskiem **Start** uruchamiany jest proces wyznaczania warunków. Pasek postępu umieszczony powyżej przycisków informuje o postępie w realizacje obliczeń. Obliczenia mogą zostać przerwane za pomocą przycisku przerwij (w wyniku przerwania wyznaczone rozwiązania nie zostaną zapisane do systemu). Po ukończeniu obliczeń w oknie (kolejno od góry) prezentowane są informacje: współczynnika nasycenia (W), liczby wyznaczonych warunków, czasu kompilacji, czasu wyznaczania wszystkich warunków, czasu uzyskania pierwszego rozwiązania (CPW). Wartości czasów wyrażone są w sekundach. Otrzymane rozwiązania automatycznie dodawane są do systemu.



Rys. B.14. Okno Warunki wystarczające

W celu uruchomienia procesu wyznaczania harmonogramów pracy poszczególnych procesów (jest to możliwe pod warunkiem wcześniejszego wyznaczenia warunków wystarczających), należy za pomocą funkcji **Wyznacz harmonogram** (Menu **Planowanie**

Transportu) otworzyć okno **Harmonogram**. Okno to (rysunek B.14) ma identyczny układ jak okno **Warunki wystarczające**. Przyciskiem **Start** uruchamiane są obliczenia.

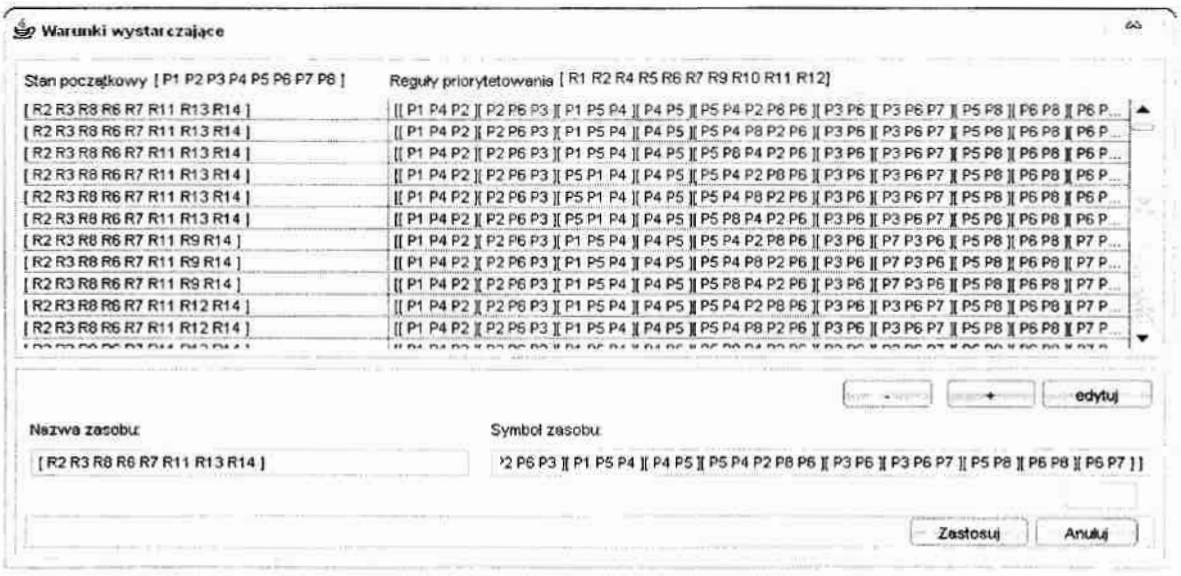


Rys. B.15. Okno Harmonogram

Po zakończeniu obliczeń w oknie prezentowe są wyniki w postaci czasu kompilacji i czasu wyznaczania rozwiązania.

Prezentacja wyników

W celu prezentacji wyznaczonych warunków wystarczających, należy za pomocą funkcji **Warunki wystarczające** (Menu **Wyniki**) otworzyć okno **Warunki wystarczające**. Okno to (rysunek B.15) zawiera listę wyznaczonych stanów początkowych oraz reguł priorytetowania.



Rys. B.16. Okno Warunki wystarczające

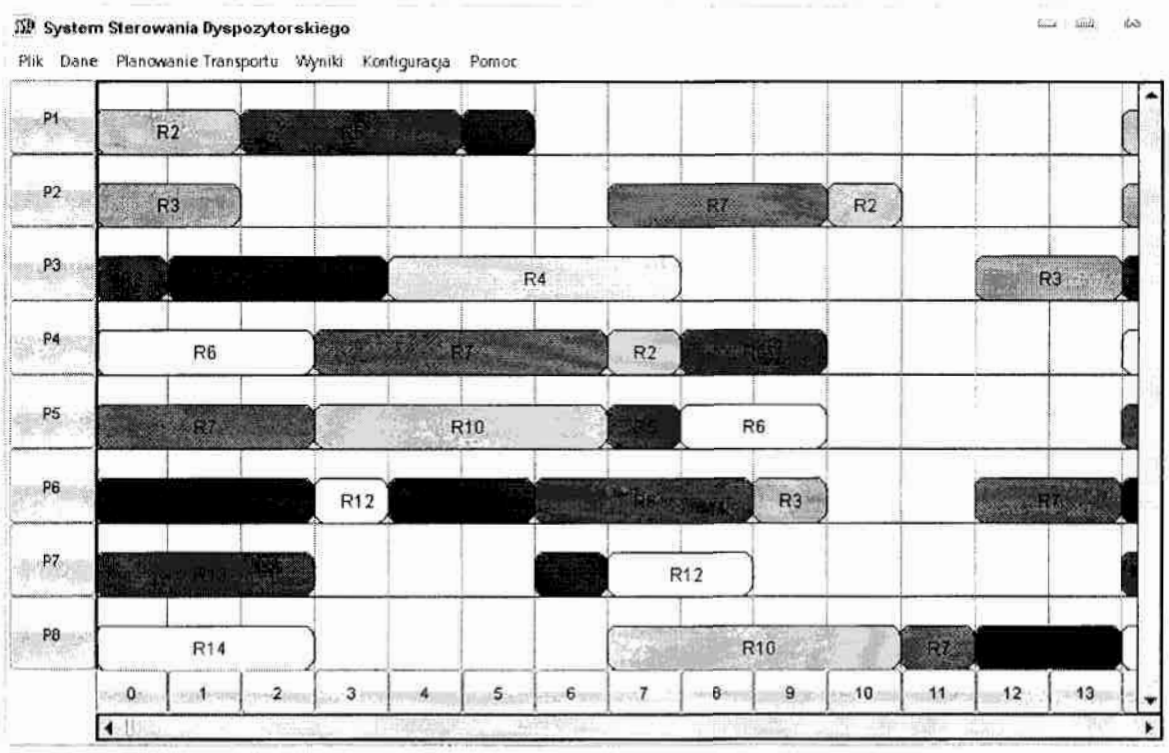
Znaczenie kolumn, widocznych w liście, jest następujące:

- ⇨ **Stan początkowy**: lista stanów początkowych S_0, której postać odpowiada wzorcowi umieszczonemu w nagłówku listy.

⇨ **Reguły priorytetowania**: lista reguł priorytetu Θ, której postać odpowiada wzorcowi umieszczonemu w nagłówku listy.

Pola **Stan początkowy** i **Reguły priorytetowania** są zbudowane w postaci list. Za pomocą tych pól możliwa jest edycja i dodawanie własnych stanów początkowych i reguł priorytetu. W listach (koniecznie w nawiasach kwadratowych) wprowadza się symbole zasobów (Stan początkowy) i symbole procesów (symbol zasobu). Przyciskiem **Zastosuj** dane zapisywane są w systemie.

W celu prezentacji wyznaczonego harmonogramu, należy za pomocą funkcji **Harmonogram dopuszczalny** (Menu **Wyniki**) wyświetlić otrzymany harmonogram. Harmonogram pojawia się w oknie głównym (rysunek B.16), stanowi on graficzną reprezentację realizacji procesów. Oś pozioma jest osią czasu (umowne jednostki czasu), oś pionowa jest osią procesów. Procesy i zasoby są oznaczane zgodnie z wprowadzonymi symbolami tych wielkości.



Legenda:

R12 - graficzna reprezentacja czasu pracy procesu na zasobie R_{12},

Pj - proces P_j.

Rys. B.17. Harmonogram realizacji poszczególnych procesów

W przypadku gdy nie istnieje harmonogram spełniający zadane ograniczenia pojawia się okno informujące o braku rozwiązania

Rys. B.18. Informacja o braku rozwiązania

Konfiguracja

Konfiguracja sytemu polega na określeniu wartości dwóch podstawowych parametrów: horyzontu oraz maksymalnej liczby wyznaczanych warunków wystarczających. Horyzont określa dziedziny zmiennych określających terminy rozpoczęcia operacji. Zbyt niska wartość horyzontu może spowodować brak odpowiedzi, nawet jeśli rozwiązanie istnieje, zbyt wysoka wartość może znacznie zwiększyć czas poszukiwania rozwiązania. Maksymalna liczba warunków określa liczbę warunków, po której proces wyznaczania warunków wystarczających zostaje wstrzymany.

W celu zmiany tych parametrów, należy za pomocą funkcji **Opcje** (Menu **Konfiguracja**) otworzyć okno **Harmonogram**. Okno to (rysunek B.18) zawiera dwa pola tekstowe: pole do wprowadzania wartości horyzontu (domyślna wartość 40), pole maksymalnej liczby warunków (domyślna wartość -1). Wartość -1 w polu maksymalnej liczby warunków oznacza, że poszukiwane są wszystkie warunki wystarczające.

Rys. B.19. Okno Konfiguracja

Dodatek C. Eksperymenty porównawcze

W oparciu o zaproponowaną strategię przeszukiwania rozwiązań dedykowanych przeprowadzone zostały eksperymenty komputerowe, których celem było wyznaczanie zbiorów S_{u1} i S_{u2} dla trzech przykładowych reprezentacji wiedzy KB_1=<U,W,Y, Re_1>, KB_2 =<U,W,Y, Re_1)>, KB_3=<U,W,Y, Re_2>. W skład każdej reprezentacji wiedzy KB_i wchodziła relacja Re_i, której postać zależała od zbioru faktów $F_i(u,w,y)$ = $\{F_{i1}(u,w,y)$, $F_{i2}(u,w,y)$, ...,$F_{i55}(u,w,y)\}$ zawierającego 55 faktów opisanych na 84 zmiennych binarnych u, w, y (zmienne mogą przyjmować wartości ze zbioru $\{0,1\}$: U = W = Y = $\{0,1\}$). Rozważane zestawy reprezentacji wiedzy nie mały żadnej interpretacji praktycznej, wygenerowane zostały w sposób losowy. Każdy z 55 faktów był zdaniem opisującym relacje na poziomie logicznym (z wykorzystaniem operatorów: $\vee$, $\wedge$, $\Rightarrow$, $\neg$, $\Leftrightarrow$) między 8 zmiennymi. Wyróżniono 14 zmiennych wejściowych tworzących zbiór: u = $\{u_1, u_2, \ldots, u_{14}\}$, 56 zmiennych pomocniczych: w = $\{w_1, w_2, \ldots, w_{56}\}$ i 14 zmiennych wyjściowych: y = $\{y_1, y_2, \ldots, y_{14}\}$. Fakt wyjściowy $Fy(y)$ stanowiło jedno zdanie logiczne opisujące relację między wszystkimi zmiennymi wyjściowymi.

Wyznaczenie zbiorów S_{u1} i S_{u2} prowadzi do rozwiązania odpowiednich problemów spełniania ograniczeń PSO_{Su1}, PSO_{Su2}, opisanych zależnościami (3.13) i (3.14). Implementacja problemów oraz eksperymenty przeprowadzono w środowisku programowania z ograniczeniami **Oz Mozart**. Wyniki przeprowadzonych eksperymentów przedstawiono w tabeli C.1.

Tab.C.1. Otrzymane wyniki w przypadku rozwiązania problemów PSO_{Su1}, PSO_{Su2}.

KB	PSO_{Su1}		PSO_{Su2}	
	Liczba elementów zbioru S_{u1}	Liczba kroków obliczeniowych	Liczba elementów zbioru S_{u1}	Liczba kroków obliczeniowych
KB_1	256	15360	320	22104
KB_2	63	3843	48	3780
KB_3	3881	232860	1502	178979

* obliczenia przeprowadzono na komputerze Pentium III 600MHz, RAM 256 MB.

Rozważone problemy PSO_{Su1}, PSO_{Su2}, zostały rozwiązane przy użyciu dwuetapowej strategii przeszukiwania rozwiązań dedykowanych. Liczba kroków obliczeniowych nie przekroczyła Z = 250000, przy czym rozmiar drzewa potencjalnych rozwiązań wynosi $R_T = 3{,}8 \cdot 10^{25}$. Stosowanie technik programowania z ograniczeniami CP, oraz zaproponowanej

strategii zaowocowało w najgorszym przypadku zyskiem $6{,}5 \cdot 10^{-21}$ w stosunku do przeszukiwania wszystkich węzłów drzewa potencjalnych rozwiązań.

Dla metody logiczno-algebraicznej znane są, rekurencyjne algorytmy generacji zbiorów S_{u1} i S_{u2} z wykorzystaniem dekompozycji [38], [27]. Zastosowanie dekompozycji prowadzi do znaczenie efektywniejszego, rekurencyjnego algorytmu rozwiązywania problemu decyzyjnego niż podejścia oparte na przeglądzie zupełnym. Dla analizowanych reprezentacji wiedzy zostało przeprowadzone porównanie podejścia opartego na technikach programowania z ograniczeniami i dwuetapowej strategii przeszukiwania, z metodą dekompozycji. W tym celu dla zadanych reprezentacji wiedzy wyznaczony został zbiór S_{u1}. Wyniki porównawcze w postaci czasów uzyskania odpowiedzi zostały przedstawione w tabeli C.2.

W przypadku technik programowania z ograniczeniami otrzymano w każdym przypadku rozwiązanie lepsze od metody rekurencyjnej.

Tab.C.2. Czasy wyznaczania zbioru S_{u1} dla procedury rekurencyjnej i technik programowania z ograniczeniami.

KB	Dwuetapowa strategia *CP* [s]	Procedura rekurencyjna [s]
KB_1	11	123
KB_2	1,7	87
KB_3	120	173

* obliczenia przeprowadzono na komputerze Pentium III 600MHz, RAM 256 M