Raport Badawczy
Research Report

RB/20/2004

Programowanie ograniczeń w systemach wspomagania decyzji MŚP

Z. Banaszak, K. Bzdyra

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Zdzisław Bubnicki

Warszawa 2004

Programowanie ograniczeń w systemach wspomagania decyzji MŚP

Zbigniew Banaszak[1)], Krzysztof Bzdyra[2)]
[1)] *Pracownia Systemów Wiedzy i Sztucznej Inteligencji IBS PAN, ul Podwale 75, Wroclaw*
[2)] *Wydział Elektroniki, Politechnika Koszalińska, ul. Śniadeckich 2, Koszalin*

Elementem warunkującym konkurencyjność przedsiębiorstwa na rynku konsumenta jest jego zdolność do podejmowania szybkich i trafnych decyzji związanych z bilansowaniem potrzeb klienta i możliwości producenta. Szczególnie często występujące w małych i średnich przedsiębiorstwach *(MŚP)* problemy decyzyjne związane są z przyjęciem do realizacji nowego zlecenia produkcyjnego. Poszukiwane jest zwykle pierwsze z rozwiązań spełniających zbiór ograniczeń wiążących zmienne decyzyjne specyfikujące możliwości producenta, zmienne charakteryzujące warunki realizacji zlecenia, a także zmienne decyzyjne układu producent-konsument. Podejmowane decyzje formułowane są zwykle w postaci problemu zgodności ograniczeń *(ang. Constraint Satisfaction Problem (CSP))*, dla którego opracowano wiele języków programowania ograniczeń *(ang. Constraint Programming (CP))*, w szczególności programowania w logice ograniczeń *(ang. Constraint Logic Programming (CLP))*. Deklaratywny charakter języków *CP* oraz wysoka efektywność implementujących je pakietów wspomagania decyzji, stwarza atrakcyjną (umożliwiającą pracę w trybie on-line) alternatywę wobec aktualnie dostępnych,systemów komputerowo zintegrowanego zarządzania. W przedstawionym kontekście, celem pracy jest wprowadzenie w problematykę programowaniem ograniczeń oraz przedstawienie możliwości jego wykorzystania w wybranych obszarach inżynierii wspomagania decyzji, w szczególności odnoszących się do *MŚP*.

Słowa kluczowe: programowanie ograniczeń, usuwanie niezgodności, systemy wspomagania decyzji

1. Wprowadzenie

Jednym z ważniejszych elementów warunkujących utrzymanie się przedsiębiorstwa na rynku konsumenta jest umiejętność oceny potrzeb rynku oraz szybkiej reakcji na te potrzeby. Łatwo zauważyć, że konkurujące przedsiębiorstwa mają dzisiaj dostęp do tych samych środków produkcji, tak samo wykształconej kadry, a także natrafiają na te same wymagania klientów. O ich sukcesie decyduje zatem zdolność do podejmowania szybkich i trafnych decyzji, decyzji związanych m.in. z bilansowaniem się potrzeb klienta i możliwości producenta.

Typowym problemem, rozwiązywanym zwykle w trybie na bieżąco, jest problem decyzyjny związany z podjęciem się lub nie realizacji nowego zlecenia produkcyjnego. Problem ten, szczególnie często występujący w małych i średnich przedsiębiorstwach (***MŚP***), specyfikuje, bardzo liczny zazwyczaj, zbiór zmiennych decyzyjnych o wartościach należących do skończonych dziedzin dyskretnych; zmiennych różnej natury i charakteru, od okresów dostępności zasobów począwszy, poprzez wielkości serii produkcyjnych i transportowych, na terminach i cenach odbioru poszczególnych partii zlecenia skończywszy. Poszukiwanym rozwiązaniem problemu jest zwykle jedno z rozwiązań dopuszczalnych, tzn. rozwiązań spełniających zbiór ograniczeń wiążących tak zmienne decyzyjne opisujące możliwości producenta, jak i zmienne charakteryzujące warunki realizacji zlecenia, a także ograniczeń wiążących niektóre zmienne decyzyjne charakteryzujące układ producent-konsument.

Przedstawione wyżej sformułowanie pokrywa się ze sformułowaniem tzw. problemu zgodności ograniczeń *(ang. Constraint Satisfaction Problem (CSP))*, dla którego opracowano wiele języków programowania ograniczeń *(ang. constraint programming (CP))*, w tym również programowania w logice ograniczeń *(ang. constraint Logic Programming (CLP))*. Deklaratywny charakter języków ***CP*** oraz wysoka efektywność w rozwiązywaniu dużych problemów

kombinatorycznych, implementujących je pakietów wspomagania decyzji, stwarza atrakcyjną alternatywę wobec aktualnie dostępnych, opartych na technikach badań operacyjnych, systemów komputerowo zintegrowanego zarządzania.

W przedstawionym kontekście, ważną z perspektywy *MŚP*, jest ocena możliwości implementacji technik programowania ograniczeń w pakietach oprogramowania wspomagającego zarządzanie przedsiębiorstwem. Pożądane aplikacje winny uwzględniać specyfikę *MŚP*, tj. istniejące ograniczenia finansowe (limitujące możliwości zakupu i utrzymania oprogramowania) i kadrowe (limitowane możliwości zatrudnienia licznego i/lub wysokokwalifikowanego personelu), a także potrzeby związane m.in. z częstym podejmowaniem (w trybie na bieżąco) standardowych decyzji (zwykle związanych z zarządzaniem w środowisku wielo-projektowym) oraz skracaniem łańcucha logistycznego łączącego decydenta z wspierającą go bazą danych (tzn. ograniczaniem liczby ogniw pośredniczących typu, operator/programista systemu, analityk, konsultant, itp.).

Celem pracy jest przedstawienie podstawowych potrzeb i ograniczeń towarzyszących wspomaganiu decyzji w *MŚP*, wprowadzenie w podstawowe zagadnienia związane z programowaniem ograniczeń oraz przedstawienie możliwości jego wykorzystania w wybranych obszarach inżynierii wspomagania decyzji zorientowanych na *MŚP*.

2. Wspomaganie decyzji w MŚP

2.1 MŚP –potrzeby i ograniczenia

Potrzeby związane ze wspomaganiem podejmowania decyzji obejmują całą strefę działalności przedsiębiorstwa od planowania strategicznego, planowania sprzedaży i produkcji, poprzez techniczne przygotowanie produkcji, bilansowanie mocy produkcyjnych, realizację

zaopatrzenia, produkcję i kontrolę planu wykonania produkcji, finanse i księgowość, zarządzanie kadrami i płacami, na realizacji i kontroli planu sprzedaży skończywszy [1]. W zależności od skali przedsiębiorstwa (jak np. przedsiębiorstwo rodzinne dysponujące dwoma lub trzema samochodami dostawczymi, małe przedsiębiorstwo dysponujące siecią hurtowni i sklepów, średnie przedsiębiorstwo produkujące elementy hydrauliki siłowej) potrzeby te są bardzo zróżnicowane. Wyrażają się one zarówno w skali i funkcjonalności poszukiwanego oprogramowania (pojedyncze aplikacje, pakiety aplikacji, pakiety umożliwiające samodzielną konfigurację parametrów oprogramowania), jak i związanych z tym platform informatycznych (jednostanowiskowe, wielostanowiskowe, pracujące w systemie DOS i/lub WINDOWS, sieciowe, itp.).

Badania przeprowadzone wśród ***MŚP*** [13], jako najistotniejsze, podkreślają potrzeby związane ze zwiększeniem udziału oraz osiągnięciem przewagi rynkowej. Badania te potwierdzają wyniki, niezależnie od prowadzonych analiz [11], wskazujące na oczekiwania tej grupy przedsiębiorstw. Oczekiwania związane z wdrożeniem technologii informacyjnych *(ang. information technology (IT))* wiążą się z potrzebami zaspokojenia potrzeb:

- zewnętrznych, związanych z przetwarzaniem danych na potrzeby kontroli i monitorowania, opracowywaniem statystyk, archiwizacją danych, itp.
- wewnętrznych, związanych z przetwarzaniem danych na potrzeby rachunkowości zarządczej, opracowywaniem statystyk, zbieraniem danych dla potrzeb planowania i sterowania dyspozytorskiego, itp.

Potrzeby te obejmują również różne obszary funkcjonowania przedsiębiorstwa: od automatyzacji prac biurowych począwszy, przez obsługę zleceń kadrowo-płacowych i finansowo-księgowych oraz techniczne przygotowanie produkcji, po zarządzanie przedsiębiorstwem, włączając w to m.in. *Customer Relationship Management (CRM), Total Quality Management (TQM)* oraz *Supply Chine Management (SCM)*. Warto podkreślić, że

pierwszoplanowymi w przedstawionej hierarchii potrzeb są potrzeby o charakterze administracyjnym związane ze sprawnym obiegiem dokumentów. W dalszej kolejności plasują się te związane z planowaniem produkcji (np. podejmowaniem nowych zleceń) czy też planowaniem strategicznym (np. planowaniem przebranżowienia się firmy). Nie tłumaczy to wcale stosunkowo nikłego zainteresowania ***MŚP*** pakietami typu: *MS Project, ARENA*, czy *Taylor*. Tym bardziej, że większość ***MŚP*** funkcjonując w środowisku wieloprojektowym wymaga wsparcia w procesie zarządzania przez projekt [5].

Zaspokojenie w/w potrzeb, warunkują wysokie koszty ***IT***. Na koszty te, oprócz związanych z zakupem sprzętu, składają się koszty zakupu licencji i utrzymania oprogramowania (np. serwis) oraz koszty wynajęcia i utrzymania (np. zatrudnienia specjalnego pracownika) odpowiedniego stanowiska.

Przykładowe koszty oferowanego oprogramowania mieszczą się od ok. 500 PLN (pracujący pod ***DOS*** zestaw *Rachmistrz – mikroSubiekt*, oferowany przez firmę *Insert* [12]) do około, 20 tys. PLN. Oferty takie obejmują zazwyczaj jednak tylko pojedyncze moduły i nie uwzględniają kosztów licencji, jak również, związanych z nią, usług konsultingowych. Szacunkowe koszty wdrożenia systemu klasy komputerowo zintegrowanego zarządzania *(ang. Enterprise Resource Planning (ERP))* w ***MŚP***, obejmującego kilkadziesiąt stanowisk (20-30 licencji) szacuje się na poziomie 400 tys. PLN [15]. Cena ta obejmuje koszty sprzętu, oprogramowania oraz usługi konsultingowe. Do wymienionych dochodzą również koszty utrzymania systemu, mieszczące się zwykle w przedziale 5 – 10% ceny licencji.

Oprócz wymienionych, kosztowych ograniczeń, limitujących inwestycje informatyczne ***MŚP***, warto również wskazać na ryzyko związane z wdrażaniem przedsięwzięcia informatycznego. Powszechnie znanym faktem jest, że ponad 70% wszystkich projektów informatycznych przekracza planowany budżet, bądź termin, bądź też jednocześnie oba te założenia. Oznacza to, że poszukiwane przez ***MŚP*** oprogramowanie winno być nie tylko funkcjonalne i tanie, ale

przede wszystkim winno gwarantować (w zakładanym okresie) zwrot poniesionych nakładów.

2.2 Oprogramowanie dla MŚP

Ze względu na istniejąca niestabilność regulacji gospodarczych (pociągającą konieczność wprowadzania częstych zmian – średnio raz na kwartał [12]), jak również zapóźnienie wykorzystywanych platform informatycznych (ok. 30% oprogramowania dla ***MŚP*** pracuje pod ***DOS***-em [12]), dostawcami oprogramowania dla ***MŚP*** są w zasadzie jego krajowi producenci. Przedstawiana oferta obejmuje zestawy osobnych aplikacji (np. firmy *Insert, Wa-Pro*), które mogą być konfigurowane w zintegrowane systemy wspierające zarządzanie firmą, jak również klasyczne pakiety (np. firmy *CDN, Matrix.pl*), integrujące od kilku do kilkudziesięciu aplikacji, tzn. oprogramowanie umożliwiające samodzielne konfigurowanie zestawów odpowiadających profilowi danej firmy.

Interesującą, w przedstawionym kontekście propozycją, jest oferta firmy *Novell* zwracającej uwagę na fakt, iż o efektywności wdrożenia danego projektu informatycznego (oddzielnej aplikacji czy pakietu) decyduje jego środowisko, tzn. bezawaryjnie działający system sieciowy wraz aplikacjami narzędziowymi. Firma ta, bezpłatnie oferuje pakiety systemu sieci komputerowej *Nowell Small Busines Suite 6PL Starter Pack* (wersja dla trzech użytkowników).

Bogata oferta systemów ***ERP*** (prezentowana przez takie firmy jak: *SAP, ORACLE, IFS, BAAN*) skierowana jest głównie dla dużych przedsiębiorstw. Cena oferowanych produktów przekracza zwykle możliwości finansowe ***MŚP***. Dostępne oprogramowanie nie uwzględnia również specyfiki ***MŚP***, związanej np. z ograniczonym zakresem wspomaganych funkcji czy też z obowiązującą w firmie strukturą obiegu informacji. Warto zauważyć, że zakres możliwości praktycznego wykorzystania, wbudowanych w nich, modułów planowania

produkcji ograniczają implementowane w tych modułach metody badań operacyjnych (jak np. metody *CPM, PERT* w *MS Project*). W szczególności oznacza to ograniczenie zakresu podejmowanych decyzji do problemów, o znanych, wcześniej opracowanych algorytmach rozwiązania.

Alternatywne, wobec konieczności zakupu odpowiedniego systemu wspomagania, jest podejście bazujące na spostrzeżeniu, że modularne w swej strukturze systemy ***ERP*** korzystają z zintegrowanych baz danych zawierających dane opisujące wszystkie obszary działalności przedsiębiorstwa. Przykładem tego są rozwiązania oparte na relacyjnych bazach danych, np. systemu *ORACLE*, bazach posiadających wbudowane mechanizmy (deklaratywne więzy spójności i proceduralne – tzw. wyzwalacze bazy danych) zapewniające integralność danych. Łatwo zauważyć, że mechanizmy te umożliwiają oprogramowanie podstawowych struktur i procedur przetwarzania występujących w systemach ***ERP***, np. w systemach klasy ***MRP II*** [14]. Programowanie tego typu dedykowanych rozwiązań przyspiesza przetwarzanie danych, a dla ***MŚP*** stanowi atrakcyjną alternatywę wobec komercyjnie dostępnych pakietów.

Kolejne, alternatywne rozwiązanie niesie koncepcja outsourcing'u umożliwiającego korzystanie z usług producentów oprogramowania zintegrowanych baz danych i/lub systemów zintegrowanego zarządzania. Rozwiązanie takie umożliwia znaczne ograniczenie kosztów działalności podstawowej przedsiębiorstwa. Pozwala również uniknąć ryzyka związanego z wdrażaniem projektu informatycznego.

Reasumując, dostępne oprogramowanie umożliwia zaspokojenie potrzeb związanych ze standardowymi (najczęściej o charakterze administracyjno-księgowym) decyzjami występującymi w działalności ***MŚP***. Brak jest rozwiązań wspierających te przedsiębiorstwa przy decyzjach związanych z zarządzaniem przez projekt [5], tzn. rozwiązań umożliwiających szybkie i tanie wariantowanie alternatywnych sposobów przyjmowania i realizacji zleceń produkcyjnych, planowania przepływów produkcji i usług, itp.

3. Programowanie ograniczeń

Programowanie ograniczeń jest obszarem technologii oprogramowania bazującym na specyfikacji ograniczeń zmiennych decyzyjnych rozwiązywanych problemów. Istotną cechą ograniczeń (relacji) stanowi ich deklaratywny charakter. Oznacza to, że ograniczenia specyfikują jedynie postać wymaganych relacji, nie podają natomiast sposobu gwarantującego ich zachodzenie. W tym kontekście, specyfikację problemu stanowią zbiory zmiennych i ich dziedzin oraz zbiór ograniczeń wiążących wybrane zmienne decyzyjne. Poszukiwanym rozwiązaniem jest zbiór wartości zmiennych spełniających przyjęte ograniczenia.

3.1 Problem zgodności ograniczeń

W zależności od charakteru dziedzin zmiennych decyzyjnych, programowanie ograniczeń obejmuje badania: związane z oceną zgodności ograniczeń *(ang. constraint satisfaction)* dla zmiennych, wartości których należą do skończonych dziedzin dyskretnych oraz rozwiązywaniem ograniczeń *(ang. constraint solving)* dla zmiennych należących do dziedzin nieskończonych. Z dostępnych ocen [4] wynika, że ponad 95% wszystkich problemów decyzyjnych w obszarze produkcji i usług należy do pierwszej z ww. kategorii, tzn. problemów zgodności ograniczeń *(ang. Constraint Satisfaction Problem (CSP))*.

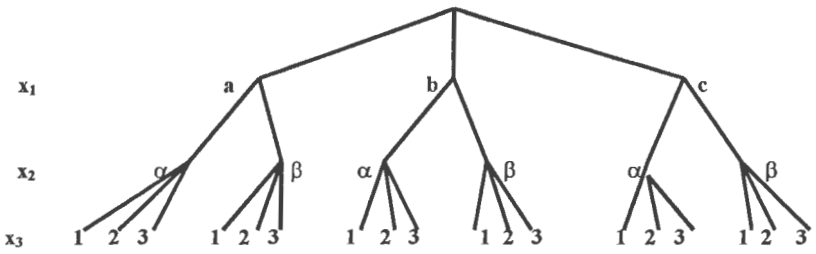
Problem zgodności ograniczeń (*PZO*). Dany jest skończony zbiór zmiennych $X = \{x_1, x_2, ..., x_n\}$, rodzina dziedzin zmiennych $D = \{D_i \mid D_i = [d_{i1}, d_{i2}, ..., d_{ij}, ..., d_{im}], i = 1..n\}$ oraz skończony zbiór ograniczeń $C = \{C_i \mid i = 1..L\}$ limitujących wartości zmiennych decyzyjnych. Poszukiwane jest rozwiązanie bądź to dopuszczalne, tzn. rozwiązanie w którym wartości wszystkich zmiennych spełniają wszystkie ograniczenia (jedno – najwcześniej uzyskane, bądź

też wszystkie możliwe), bądź też rozwiązanie optymalne (w ogólności zbiór rozwiązań) ekstremalizujące funkcję celu określoną na wybranym podzbiorze zmiennych decyzyjnych.

W poszukiwaniu rozwiązania optymalnego, tzn. rozważając tzw. optymalizacyjny problem zgodności ograniczeń *(ang. Constraint Satisfaction Optimization problem (CSOP))* traktowany jako rozszerzenie ***PZO*** o funkcje celu, wykorzystywane są metody podziału i ograniczeń bazujące na heurystycznych funkcjach oceny przestrzeni rozwiązań dopuszczalnych (lub tzw. częściowych przyporządkowań – związanych z przyporządkowaniami wartości tylko do pewnej części wszystkich zmiennych decyzyjnych) .

Rozwiązanie problemu *PZO* uzyskiwane jest w wyniku systematycznego przeszukiwania możliwych przyporządkowań wartości zmiennych decyzyjnych. Wykorzystywane metody poszukiwania rozwiązań dzielą się na te, w których przeszukiwana jest cała przestrzeń wszystkich możliwych przyporządkowań (wykorzystywane są tutaj techniki przeglądu zupełnego (metody stochastyczne) oraz na te, w których przeszukiwana jest tylko część tej przestrzeni.

Celem ilustracji rozważmy problem ***PZO*** określony przez zmienne decyzyjne x_1, x_2 *i* x_3 ich dziedziny x_1 = [a,b,c], x_2 = [α,β] x_3 = [,2,3]oraz zbiór ograniczeń $C = \{C_1, C_2, C_3\}$. Przestrzeń wszystkich rozwiązań ilustruje Rys. 1 przedstawiający drzewo wszystkich potencjalnych podstawień wartości zmiennych decyzyjnych.



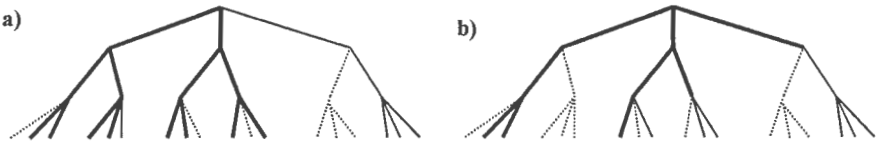
Rys. 1 Drzewo wszystkich potencjalnych podstawień wartości zmiennych decyzyjnych.

Metody systematycznego przeszukiwania bazują na dwóch podejściach związanych z: przeszukiwaniem przestrzeni potencjalnych podstawień wartości zmiennych decyzyjnych – metoda „generuj-i-testuj" *(ang. generate-and-test (GT))* oraz przeszukiwaniem z nawrotami *(ang. backtracking (BT))*. Druga z nich stanowi swoiste rozszerzenie pierwszej – w przypadku gdy aktualnie wygenerowane, częściowe podstawienie zmiennych jest sprzeczne z ograniczeniami, następuje powrót do bezpośrednio poprzedzającego częściowego podstawienia zmiennych, co pozwala ograniczyć liczbę kolejnych generacji i sprawdzeń. Oba podejścia łączy stochastyczny charakter przeszukiwania przestrzeni podstawień – podstawienia wartości zmiennych generowane są losowo.

Łatwo zauważyć, że złożoność obliczeniowa algorytmu przeglądu zupełnego wynosi $O(d^h)$, gdzie $d = max\{|D_i|\}$, $h = |X|$. Oznacza to, że w przypadku wielu spotykanych w praktyce sytuacji, nawet przyjmując, że 90% potencjalnych rozwiązań odrzucanych jest przez przyjęte ograniczenia, nie jest możliwe korzystanie z metody przeglądu zupełnego. Jako alternatywa pozostają więc metody przeglądu niepełnego, metody wykorzystujące różne heurystyki redukcji przeszukiwania drzewa podstawień (stanowiące na ogół uogólnienie szeregu, wcześniej przeprowadzonych, eksperymentów).

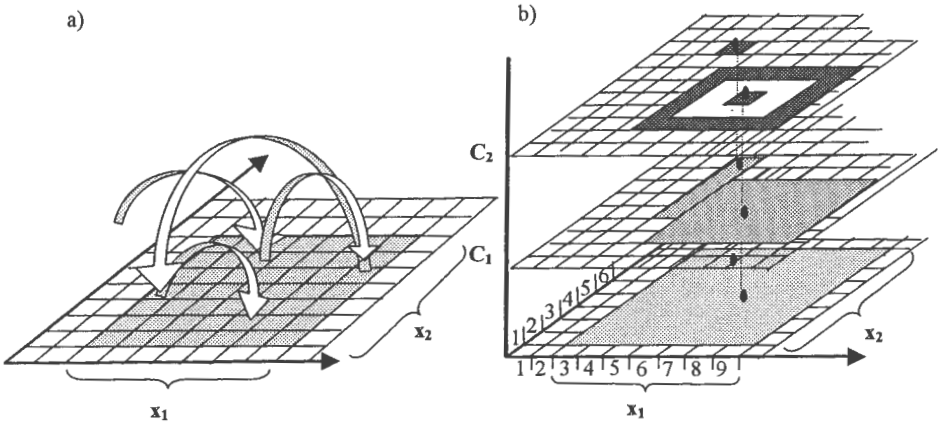
Przykładem **metod przeglądu niepełnego**, tzn. metod ograniczających przeszukiwanie przestrzeni potencjalnych podstawień wartości zmiennych decyzyjnych są, bazujące na strategii przeszukiwania „najpierw w głąb" (ang. depth-first), metoda „iteracyjnego poszerzania" *(ang. Iterative Broadening (IB))* oraz metoda „przeszukiwania ograniczonej liczby podstawień" *(ang. Limited Assignment Number Search (LAN))* [3]. Pierwsza z nich ogranicza liczbę przeszukiwanych, w każdym wierzchołku, alternatyw, do wartości tzw. „limitu szerokości" *(ang. breadth limit)*. Ilustrację metody ***IB*** dla „limitu szerokości" równego 2, przedstawia Rys. 2 a). Złożoność obliczeniową tej metody opisuje funkcja wykładnicza.

Drugą z omawianych metod charakteryzuje parametr ograniczający liczbę dopuszczalnych podstawień każdej ze zmiennych decyzyjnych. Złożoność obliczeniową tej metody opisuje funkcja ***O(nl)*** gdzie n – liczba zmiennych, l – parametr limitujący maksymalną liczbę podstawień każdej ze zmiennych. Działanie metody, dla $l = 3$, ilustruje Rys. 2 b).



Rys. 2. Metody limitowanego przeszukiwania drzewa potencjalnych podstawień wartości zmiennych decyzyjnych. a) metoda ***IB(2)*** b) metoda ***LAN(3)***. Krawędzie pogrubione oznaczają przeszukiwane (z lewa na prawo) podstawienia wartości zmiennych, natomiast krawędzie kropkowane oznaczają podstawienia sprzeczne z ograniczeniami.

Alternatywę podejścia stochastycznego stanowią **metody usuwania niezgodności** oparte na technikach propagacji ograniczeń. Różnicę obu podejść ilustruje Rys. 3. Rysunek 3 a) akcentuje stochastyczny charakter przeszukiwania (sprawdzanie warunków ograniczeń na losowo generowanych elementach) przestrzeni $x_1 \times x_2$, podczas gdy Rys. 3 b) podkreśla selekcyjną rolę ograniczeń (początkowa „populacja" elementów przestrzeni $x_1 \times x_2$ przechodzi przez kolejne „sita" ograniczeń).



Rys. 3. Metody poszukiwania rozwiązania a) przeszukiwanie stochastyczne, b) usuwanie niezgodności – poziom „zerowy" dopuszcza wszystkie możliwe podstawienia wartości zmiennych decyzyjnych; poziom pierwszy przedstawia tylko te podstawienia, które spełniają ograniczenie C_1; poziom drugi (ostatni) tylko te zaś, które spełniają oba ograniczenia C_1 i C_2.

3.2 Usuwanie niezgodności

Istota technik usuwania niezgodności (alternatywnych w stosunku do wcześniej omawianych niedeterministycznych metod przeszukiwania) sprowadza się do usuwania, niezgodnych z ograniczeniami, wartości dziedzin zmiennych decyzyjnych. Spośród wielu technik usuwania niezgodności warto wymienić trzy podstawowe [4]: „zgodności wierzchołka" *(ang. a node consistency (NC))*, „zgodności łuku" *(ang. arc consistency (AC))* oraz „zgodności ścieżek" *(ang. path consistency (PC))*.

Według pierwszej z nich usuwane są te wartości zmiennych, które stoją w sprzeczności z ograniczeniami jednoargumentowymi *(ang. unary constraints)* zmiennych decyzyjnych. Z kolei, według techniki zgodności łuku usuwane są te wartości zmiennych, które stoją w

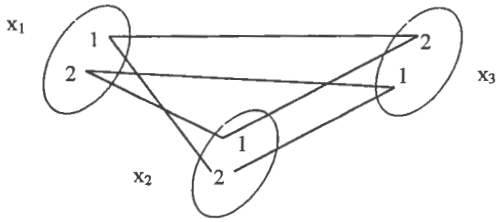
sprzeczności z ograniczeniami dwuargumentowymi (binarnymi) *(ang. binary constraints)*. Przykładowo, łuk (x_i,x_j), gdzie x_i, x_j - zmienne decyzyjne, jest zgodny w sensie ***AC*** wtedy, jeżeli dla każdej wartości v z dziedziny x_i (spełniającej ograniczenia narzucane na tą zmienną), istnieje wartość q z dziedziny x_j , taka, że wartość v i q spełniają ograniczenia binarne narzucane na zmienne decyzyjne x_i i x_j.

Zgodność ścieżek wymaga aby, dla każdej pary wartości zmiennych x_i i x_j (wyznaczającej daną ścieżkę) spełniających ograniczenie binarne, istniały również wartości zmiennych decyzyjnych należących do tej ścieżki spełniające odpowiednie ograniczenia binarne.

Przyjęte nazewnictwo, pochodzi od zwyczajowo przyjętej, graficznej reprezentacji problemu ***PZO*** – tzn. grafu, którego wierzchołki odpowiadają zmiennym, a krawędzie ograniczeniom. Reprezentacja taka wymaga sprowadzenia problemu ***PZO*** do pewnej standardowej postaci *(ang. binary **CSP**)*, zawierającej tylko jedno- i dwuargumentowe ograniczenia. Można pokazać, że dowolna postać ***problemu PZO*** jest łatwo sprowadzalna do takiej postaci standardowej [2].

Istnieje wiele algorytmów należących do technik ***AC*** i sprowadzających się do iteracyjnego sprawdzania krawędzi bądź to do uzyskania podstawienia zmiennych decyzyjnych spełniających wszystkie ograniczenia, bądź też do wyczerpania się możliwości podstawień zmiennych z danej dziedziny. Należy podkreślić, że większość algorytmów usuwania niezgodności nie zapewnia jednak pełnego przeglądu *(ang. incomplete)*. Oznacza to, że w ogólności nie wykluczają „pozostawienia" wartości sprzecznych z ograniczeniami.

Celem ilustracji rozważmy PZO zadany przez zbiór zmiennych decyzyjnych $\{x_1, x_2, x_3\}$, rodzinę dziedzin zmiennych $D_1=[1,2]$, $D_2=[1,2]$, $D_3=[1,2]$ oraz zbiór ograniczeń: $x_1 \neq x_2$, $x_2 \neq x_3$, $x_1 \neq x_3$. Rysunek 4 przestawia graficzną reprezentację binarnego problemu ***PZO***.



Rys. 4 Graficzna reprezentacja binarnego problemu ***PZO***.

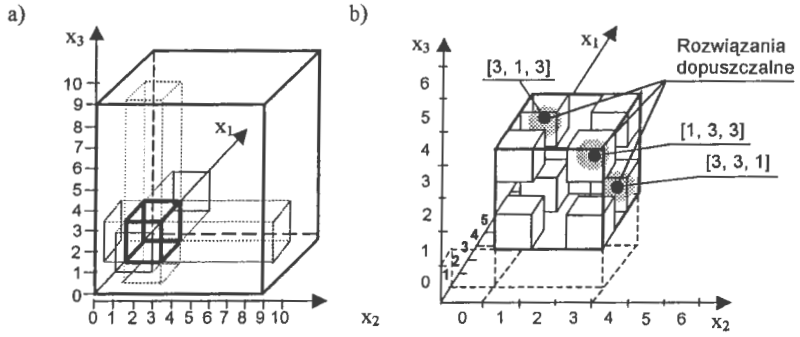
Łatwo widać, że żadne z potencjalnych ośmiu podstawień zmiennych decyzyjnych nie spełnia jednocześnie wszystkich ograniczeń, aczkolwiek poszczególne pary zmiennych spełniają wiążące je ograniczenia.

W przypadku ogólnym, w którym ograniczenia nie występują w postaci binarnej, podobna ilustracja wymagałby prezentacji przestrzennej. Celem ilustracji rozważmy ***PZO*** zadany przez zbiór zmiennych decyzyjnych $\{x_1, x_2, x_3\}$, rodzinę dziedzin zmiennych $D_1 \in [0..10]$, $D_2=[0..10]$, $D_3=[0..10]$ oraz zbiór ograniczeń:

$$x_1 + x_2 + x_3 \leq 9 \qquad (1)$$

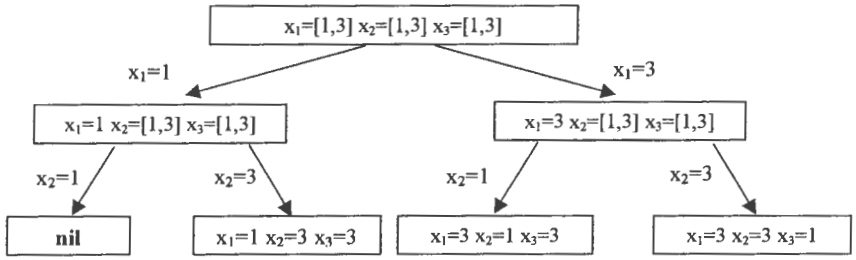
$$x_1 - x_2 - x_3 = 9 \qquad (2)$$

Można zauważyć, że przestrzeń dopuszczalnych podstawień (Rys. 5 b)), spełniająca ograniczenia (1) i (2) zawiera elementy nie będące rozwiązaniami dopuszczalnymi, tzn. elementy [1,1,3], [1,3,1], [3,1,1], [1,1,1], [3,3,3].



Rys. 5 Ograniczenia przestrzeni przeszukiwań. a) przestrzenie spełniające ograniczenie (1) oraz ograniczenia (1) i (2), b) rozwiązania dopuszczalne w przestrzeni spełniającej ograniczenia (1) i (2).

Rozwiązania dopuszczalne [1,3,3], [3,1,3], [3,3,1] uzyskane zostały przez podstawienie kolejnych wartości zmiennej x_1 (tzn. wprowadzenie dodatkowych ograniczeń), pozwalające usunąć istniejące niezgodności i zawęzić przestrzeń do zbioru rozwiązań dopuszczalnych (Rys.6).

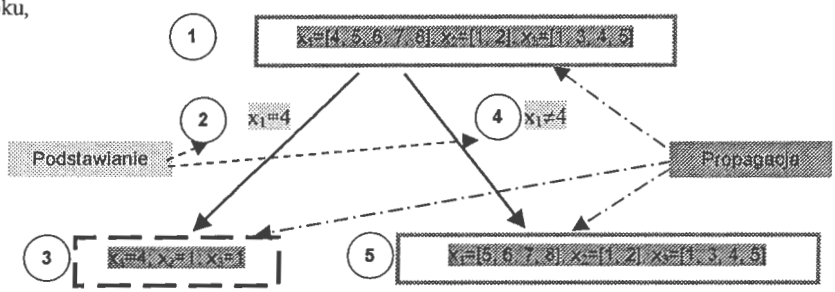


Rys. 6 Usuwanie niezgodności przez podstawianie wartości kolejnych zmiennych decyzyjnych.

3.3 Propagacja ograniczeń

Przedstawiony przykład (patrz Rys. 6) połączenia technik usuwania niezgodności i podstawiania wartości zmiennych decyzyjnych (tzn. rozszerzania zbioru ograniczeń) ilustruje możliwości budowy algorytmów poszukiwania rozwiązania problemu ***PZO*** wykorzystujące, uruchamiane na przemian, procedury usuwania niezgodności i podstawiania wartości zmiennych decyzyjnych (tzw. algorytmy propagacji ograniczeń). Procedury usuwania niezgodności odrzucają te wartości dziedzin zmiennych decyzyjnych, co do których wiadomo, że na pewno nie spełniają żadnego z ograniczeń. Procedury podstawiania umożliwiają zawężanie obszaru potencjalnych rozwiązań poprzez nadanie określonej wartości jednej ze zmiennych decyzyjnych.

Rozważany problem ***PZO*** zadany jest przez zbiór zmiennych decyzyjnych $\{x_1, x_2, x_3\}$, rodzinę dziedzin zmiennych $x_1=[1..8]$; $x_2=[1..10]$; $x_3=[1..10]$ oraz zbiór ograniczeń $x_3 \neq 2$, $x_1-x_3=3 \cdot x_2$. W wyniku pierwszej propagacji ograniczeń otrzymano rozwiązanie $x_1=[4, 5, 6, 8]$ $x_2=[1, 2]$ $x_3=[1, 3, 4, 5]$ przedstawiające wartości dziedzin poszczególnych zmiennych decyzyjnych, wartości spełniających przynajmniej jedno ograniczenie (Rys. 7). W kolejnym kroku,



Legenda:

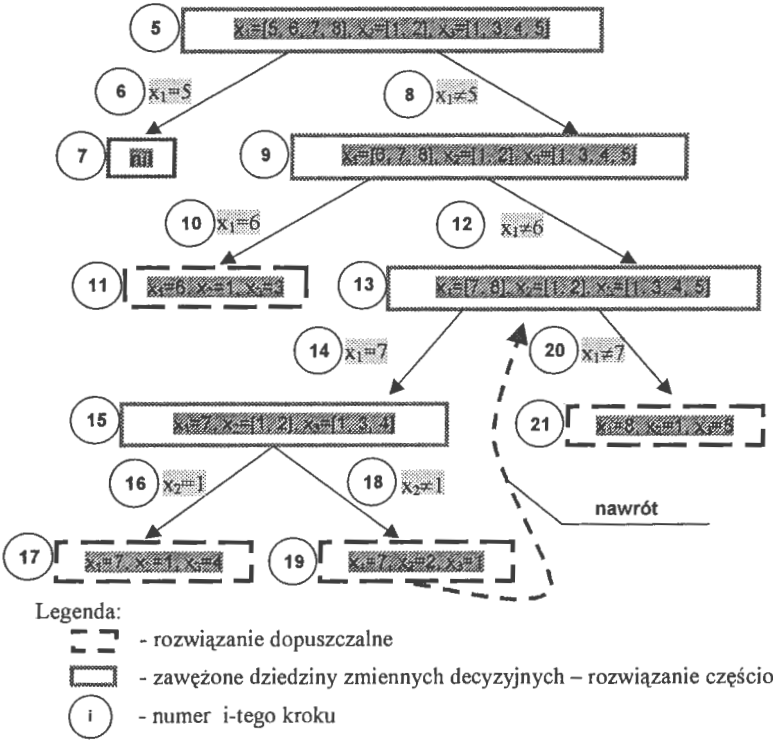
- rozwiązanie dopuszczalne
- zawężone dziedziny zmiennych decyzyjnych – rozwiązanie częściowe
- i - numer i-tego kroku

Rys. 7. Schemat propagacji ograniczeń i podstawiania– poszukiwanie pierwszego rozwiązania dopuszczalnego

dokonano podstawienia pierwszej wartości zmiennej decyzyjnej $x_1 = 4$. Podstawienie to może być interpretowane jako wprowadzenie dodatkowego ograniczenia problemu ***PZO***. Oznacza to, że w wyniku tego podstawienia wprowadzono dodatkowe ograniczenie nadające jednej ze zmiennych określoną wartość. Dla tak rozszerzonego zbioru ograniczeń, w kroku trzecim, dokonano ponownie propagacji ograniczeń. W jej wyniku dziedziny zmiennych decyzyjnych zostały zawężone do zbiorów jednoelementowych; wyznaczone zostało zatem pierwsze dopuszczalne rozwiązanie [4, 1, 1]. Jeżeli poszukiwane są wszystkie rozwiązania dopuszczalne, wówczas realizowane są następne kroki podstawiania i propagacji. Oznacza to, że dokonując, w czwartym kroku, podstawienia $x_1 \neq 4$, w kolejnym, piątym kroku, w wyniku propagacji ograniczeń, nowo-uzyskane dziedziny zmiennych nie są dziedzinami jednoelementowymi. Konieczne jest, zatem prowadzenie kolejnych etapów podstawiania i propagacji.

Schemat poszukiwań kolejnych rozwiązań dopuszczalnych przedstawia Rys. 8. Warto zwrócić uwagę na podstawienie $x_1=5$ (krok szósty). W wyniku propagacji uwzględniającej to ograniczenie, dziedziny pozostałych zmiennych decyzyjnych okazały się być zbiorami pustymi. Oznacza to, że dla podstawienia $x_1=5$ rozwiązanie dopuszczalne nie istnieje.

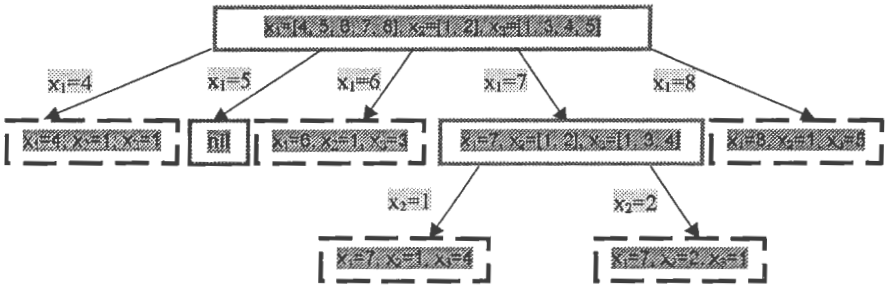
Z kolei podstawienie $x_1=7$ (krok czternasty), w kolejnej propagacji (krok piętnasty), prowadzi do rozwiązania, w którym dziedziny zmiennych decyzyjnych są więcej niż jednoelementowe. Oznacza to konieczność podstawiania wartości kolejnej zmiennej decyzyjnej $x_2=1$ (krok szesnasty). W wyniku propagacji otrzymano następne rozwiązanie dopuszczalne (krok siedemnasty).



Rys. 8. Wyszukiwanie całego zbioru rozwiązań dopuszczalnych

Kolejne podstawienie $x_2 \neq 1$ (w rozważanym przypadku $x_2=2$) również prowadzi do rozwiązania dopuszczalnego. Po kroku dziewiętnastym, nastąpił powrót (nawrót) *(ang. backtracking)* do kroku trzynastego i próby podstawienia $x_1 \neq 7$.

W praktyce korzysta się często z uproszczonej postaci drzewa przeszukiwań (Rys. 9).



Rys. 9. Alternatywny sposób przedstawienia drzewa przeszukiwań

3.4 Strategie przeszukiwania

W literaturze przedmiotu znanych jest wiele strategii przeszukiwania przestrzeni potencjalnych podstawień wartości zmiennych decyzyjnych [2], [3], [4]. O ich efektywności stanowią, wykorzystywane przez nie, strategie podstawiania. Ich efektywność zależna jest z kolei od kolejności podstawiania zmiennych decyzyjnych (np. x_1-x_2-x_3 lub x_3-x_1-x_2, itp.), wartość zmiennej decyzyjnej począwszy od której następuje podstawianie zmiennej (dolna lub górna granica dziedziny zmiennych decyzyjnych, konkretna wartość z dziedziny, itp.). W ogólności, efektywność poszczególnych strategii zależy również od specyfiki problemu ***PZO***, tzn. od struktury determinujących go danych.

W przypadku poszukiwania zbioru wszystkich rozwiązań dopuszczalnych, wynik końcowy, niezależnie od strategii, zawsze jest taki sam (w przeciwieństwie do przypadku wyznaczania pierwszego rozwiązania dopuszczalnego), czasy odpowiednich obliczeń mogą się natomiast różnić w sposób bardzo istotny.

Celem zilustrowania wpływu strategii podstawiania na czas obliczeń i jakość uzyskiwanych rozwiązań rozważamy problem ***PZO*** dekomponujący się na dwa podproblemy. Strategie podstawiania, dla każdego z tych podproblemów, są wybierane niezależnie.

Należy przewieźć partię towaru P. Do dyspozycji jest N ciężarówek, każda o ładowności L.

Zmienne decyzyjne i ich dziedziny:

LD - liczba partii transportowych na jakie zostanie podzielona partia P, LD=1..N,

T=$\{T_1, T_2,..., T_N\}$ - wielkości partii transportowych przewożonych poszczególnymi ciężarówkami. T_i=0..L dla i=1..N

Ograniczenia

$$\sum_{i=1}^{N} T_i = P \qquad (1)$$

$$\sum_{i=1}^{N} \mathrm{sgn}(T_i) = LD \tag{2}$$

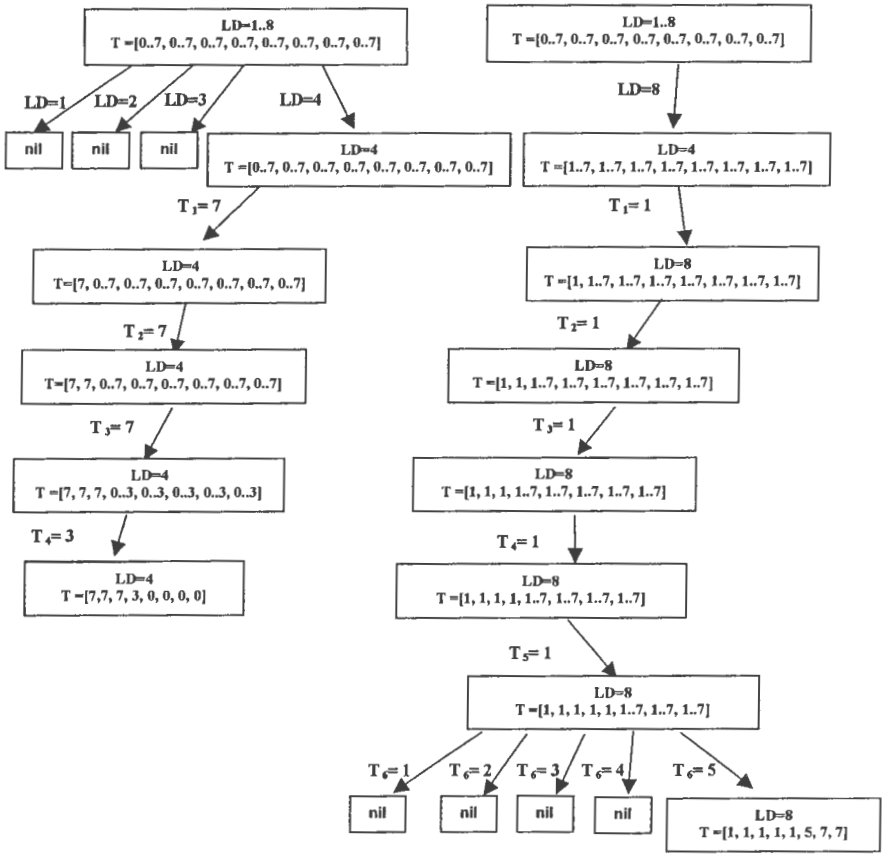
Rozważmy instancję: N=8, P=24, L=7 oraz dwie strategie podstawiania: ***value:min*** i ***value:max***. Celem eksperymentów jest wyznaczenie minimalnej liczby poszukiwań prowadzących do pierwszego rozwiązania dopuszczalnego. Ocenianie są wybrane alokacje strategii podstawiania do zmiennych decyzyjnych oraz różne kolejności realizacji strategii podstawiania.

Wyniki przeprowadzonych eksperymentów zebrano w Tabeli 1

Tabela 1 Wyniki eksperymentów

Kolejność podstawiania zmiennych decyzyjnych	alokacja strategii		Ilość kroków przeszukiwania
	LD	*T*	
$LD \rightarrow T$	***value:min***	***value:max***	8 ***(rys. 7a)***
	value:max	***value:min***	11 ***(rys. 7b)***
$T \rightarrow LD$	***value:min***	***value:max***	4 ***(rys. 8a)***
	value:max	***value:min***	7 ***(rys. 8b)***

a) b)



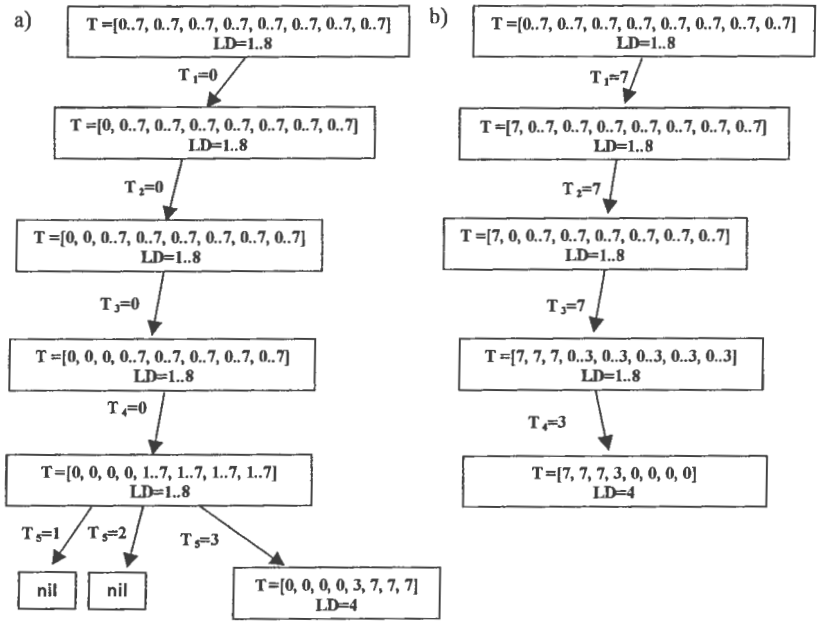
Rys.10. Drzewo poszukiwania pierwszego rozwiązania dopuszczalnego przy kolejności podstawiania zmiennych decyzyjnych $LD \rightarrow T$. a) zmiennej LD przypisano strategię ***value:min***, a zmiennej T strategię ***value:max***. b) zmiennej LD przypisano strategię ***value:max***, a zmiennej T przypisano strategię ***value:min***

Łatwo zauważyć, że przy tych samych alokacjach strategii do zmiennych decyzyjnych istotną rolę odgrywa kolejność w jakiej strategie te są realizowane. W rozważanym przypadku

kolejność, według której podstawienie zmiennej decyzyjnej *T* poprzedza podstawienie zmiennej decyzyjnej *LD*, skutkuje mniejszą liczbą kroków.

Z kolei, przy różnych kolejnościach podstawiania zmiennych decyzyjnych istotne znaczenie ma alokacja strategii podstawiania. W analizowanym przypadku, lepsze wyniki uzyskano podstawiając zmienną decyzyjną *LD* według strategii ***value:min*** .

Poszczególne fazy poszukiwań oraz uzyskane rozwiązania przedstawione zostały na Rys. 10 i Rys.11. Zbiór rozwiązań dopuszczalnych obejmuje trzy optymalne (z dokładnością do numeracji ciężarówek) rozwiązania (w sensie kryterium liczby wykorzystywanych ciężarówek). Czwarte rozwiązanie zakłada wykorzystanie 5 ciężarówek (przewożących po 1 sztuce towaru,) jednej przewożącej 5 sztuk oraz dwóch przewożących po 7 sztuk materiału.



Rys. 11. Drzewo poszukiwania pierwszego rozwiązania dopuszczalnego przy kolejności podstawiania zmiennych decyzyjnych $T \rightarrow LD$. a) zmiennej T przypisano strategię ***value:min***, a zmiennej *LD* strategię ***value:max***. b) zmiennej *T* przypisano strategię ***value:max***, a zmiennej *LD* przypisano strategię ***value:min***

Rozważając zatem tylko przypadki przedstawione na rysunkach 10 i 11, łatwo zauważyć że dobór procedury podstawiania ma istotny wpływ ma jakość rozwiązań oraz liczbę obliczeń niezbędnych do ich wyznaczenia. Przy odpowiednio dobranej strategii można otrzymać rozwiązanie optymalne już w pierwszym kroku, a co więcej można ograniczyć liczbę deklarowanych zmiennych decyzyjnych.

4. Inżynieria wspomagania decyzji

Decyzje, jak inne dobra charakteryzuje czas i koszt ich wypracowania oraz związana z nimi jakość (poziom ryzyka, terminowość ich podjęcia, itp.). O ich konkurencyjności zatem, podobnie jak ma to miejsce w przypadku innych towarów, decydują nowoczesne techniki i technologie, a w szczególności *IT*. Oznacza to, że mówiąc o racjonalnym wspomaganiu decyzji w *MŚP*, nie można pominąć aspektów inżynierii wspomagania decyzji, tzn. metod wsparcia decyzji i ich komputerowych implementacji.

4.1 Struktury danych

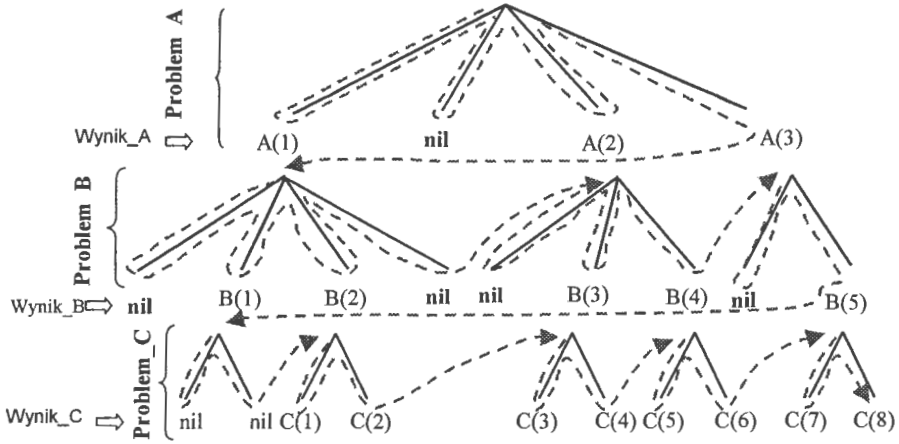
Jak już wspomniano, efektywność poszczególnych strategii poszukiwania rozwiązania zależy tak od wykorzystywanych strategii podstawiania i usuwania niezgodności, jak i od specyfiki problemu *PZO*, tzn. od struktury danych go determinujących. Spostrzeżenie to oznacza możliwość, podobnie jak ma to miejsce z regułami wyboru priorytetu, eksperymentalnego doboru (np. drogą przeprowadzania szeregu testów komputerowych) najlepszych, dla danych klas problemów *PZO*, strategii przeszukiwania.

Praktyka pokazuje, że różne strategie propagacji ograniczeń znajdują swoje różne, komputerowe implementacje. Każdy spośród dostępnych języków programowania *CP/CLP*

(np. *Prolog, CHIP, Oz, Ilog*) wymusza, charakterystyczne dla siebie sposoby specyfikacji problemu ***PZO***. Dotyczy to, w szczególności, implementacji ograniczeń nieliniowych, takich gdzie występują wartości bezwzględne, część całkowita z dzielenia (lub reszta z dzielenia), funkcje logiczne itp. Sytuacja taka powoduje, że pewne implementacje ograniczeń, nie stwarzające trudności w jednym języku, stanowią istotne ograniczenie implementacyjne w drugim, i odwrotnie. Rzutuje to oczywiście na efektywność implementowanych strategii propagacji ograniczeń.

Częstym sposobem rozwiązywania tego typu problemów jest dekomponowanie zadania na części (podproblemy). W ten sposób, rozwiązanie wcześniej rozważanego podproblemu stanowi ograniczenie przy rozwiązywaniu kolejnego podproblemu. Podejście takie pozwala wyróżnić dwie strategie przeszukiwania przestrzeni potencjalnych podstawień zmiennych decyzyjnych: strategie przeszukiwania sekwencyjnego oraz strategie przeszukiwania zagnieżdżonego *(ang. nested search)*.

Charakterystyczną cechą przeszukiwania sekwencyjnego jest to, że na każdym etapie (związanym z rozwiązywaniem danego podproblemu) poszukiwany jest zbiór rozwiązań dopuszczalnych, po czym dla każdego w ten sposób uzyskanego rozwiązania (rozszerzającego zbiór ograniczeń) poszukiwanie jest rozwiązanie kolejnego podproblemu. Przeszukiwanie to nie dopuszcza nawrotów do etapów wcześniejszych. Rys. 12 i 13 przedstawiają, odpowiednio, drzewo przeszukiwania oraz schemat implementacji metody przeszukiwania sekwencyjnego.



Rys. 12. Drzewo przeszukiwania metodą sekwencyjną

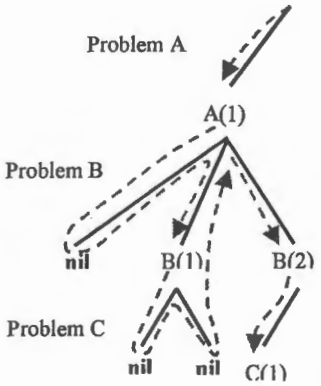
```
deklaracja danych wejściowych(Dane_We)
funkcja Problem_A (Dane)
      deklaracja zmiennych decyzyjnych
      deklaracja ograniczeń
      określenie strategii dystrybucji
koniec_funkcji
funkcja Problem_B (Dane)
      deklaracja zmiennych decyzyjnych
      deklaracja ograniczeń
      określenie strategii dystrybucji
koniec_funkcji
funkcja Problem_C (Dane)
      deklaracja zmiennych decyzyjnych
      deklaracja ograniczeń
      określenie strategii dystrybucji
koniec_funkcji
program_główny
      określenie zmiennych wejściowych
      Wynik_A=WszystkieRozwiązania(Problem_A (Dane_We))
      Wynik_B=WszystkieRozwiązania(Problem_B (Wynik_A))
      Wynik_C=WszystkieRozwiązania(Problem_C (Wynik B))
      Wynik_Końcowy={Wynik_A Wynik_B Wynik_C}
koniec_programu
```

Rys.13. Przykład implementacji programowej metody przeszukiwania sekwencyjnego

Metoda przeszukiwania zagnieżdżonego polega na poszukiwaniu pierwszego rozwiązania dopuszczalnego danego podproblemu, a po jego znalezieniu, przejściu do poszukiwania

pierwszego rozwiązania dopuszczalnego kolejnego podproblemu. Jeżeli na kolejnym etapie poszukiwań nie można znaleźć rozwiązania dopuszczalnego, wówczas następuje powrót do etapu wcześniejszego, a poszukiwanie kontynuowane jest dla kolejnego rozwiązania dopuszczalnego. Metoda ta dopuszcza nawroty, umożliwiając tym samym również wyszukanie wszystkich rozwiązań dopuszczalnych.

Rys. 14 i 15 przedstawiają, odpowiednio, drzewo przeszukiwania oraz schemat implementacji metody przeszukiwania.



Rys. 14. Drzewo przeszukiwania metodą zagnieżdżeń

```
deklaracja danych wejściowych(Dane_We)
funkcja Problem_ABC (Dane)
      deklaracja zmiennych decyzyjnych problemu A
      deklaracja ograniczeń problemu A
      określenie strategii dystrybucji
      deklaracja zmiennych decyzyjnych problemu B
      deklaracja ograniczeń problemu B
      określenie strategii dystrybucji
      deklaracja zmiennych decyzyjnych problemu C
      deklaracja ograniczeń problemu C
      określenie strategii dystrybucji
koniec_funkcji
program_główny
      określenie zmiennych wejściowych
      Wynik_Końcowy=Jedno_Rozwiazanie(Problem_ABC(Dane_We))
koniec_programu
```

Rys. 15. Przykład implementacji programowej metody przeszukiwania zagnieżdżonego.

Łatwo zauważyć, że nakłady obliczeniowe ponoszone przy rozwiązywaniu zdekomponowanego problemu ***PZO*** są większe niż ma to miejsce w przypadku problemów nie zdekomponowanych. Częstokroć jednak, tylko odpowiednia dekompozycja problemu ***PZO*** umożliwia jego implementację w dostepnych językach ***CP/CLP***.

Celem ilustracji rozważmy problem ***PZO*** dekomponujący się na podproblem A scharakteryzowany dwoma zmiennymi decyzyjnymi x_1 i x_2 oraz podproblem B opisany zmiennymi x_3 i x_4. Znane są dziedziny zmiennych x_1 =[1..10], x_2=[1..10], x_3 =[1..8] i x_4=[1..8] oraz ograniczenia:

- dla zmiennych podproblemu A

$$x_1 \neq x_2 \tag{3}$$

$$x_1+x_2=12 \tag{4}$$

- dla zmiennych podproblemu B

$$x_3 \geq 2 \cdot x_4 \tag{5}$$

$$x_3 \leq 3 \cdot x_4 \tag{6}$$

- wiążące w/w podproblemy (ograniczenie sprawdzające zgodność rozwiązań podproblemu A i B).

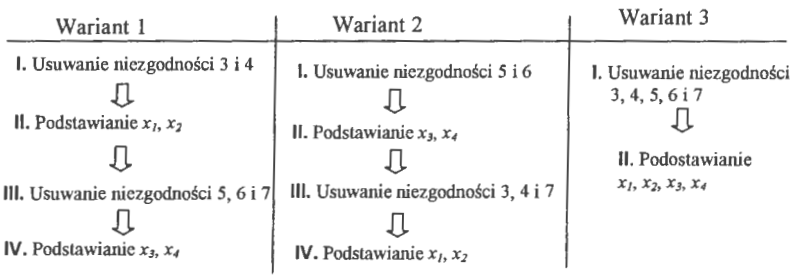
$$x_2 \cdot x_3 < x_1 \cdot x_4 \tag{7}$$

Rozważane są trzy warianty rozwiązania zadania. (Rys.16).

Zadanie dekomponowane		Zadanie nie zdekomponowane
Wariant 1	Wariant 2	Wariant 3
Znajdź rozwiązania podproblemu A spełniające ograniczenia 3 i 4 dla x_1, x_2 ⇩ Znajdź rozwiązania podproblemu B spełniające ograniczenia 5, 6 i 7 dla x_3, x_4	Znajdź rozwiązania podproblemu B spełniające ograniczenia 5 i 6 dla x_3, x_4 ⇩ Znajdź rozwiązania podproblemu A spełniające ograniczenia 3, 4 i 7 dla x_1, x_2	Znajdź rozwiązania spełniające ograniczenia 3-7 dla x_1, x_2, x_3, x_4

Rys. 16. Warianty rozwiązania rozważanego zadania

Rysunek 17 Przedstawia uproszczone schematy implementacji komputerowej poszczególnych wariantów rozwiązania problemu ***PZO***.



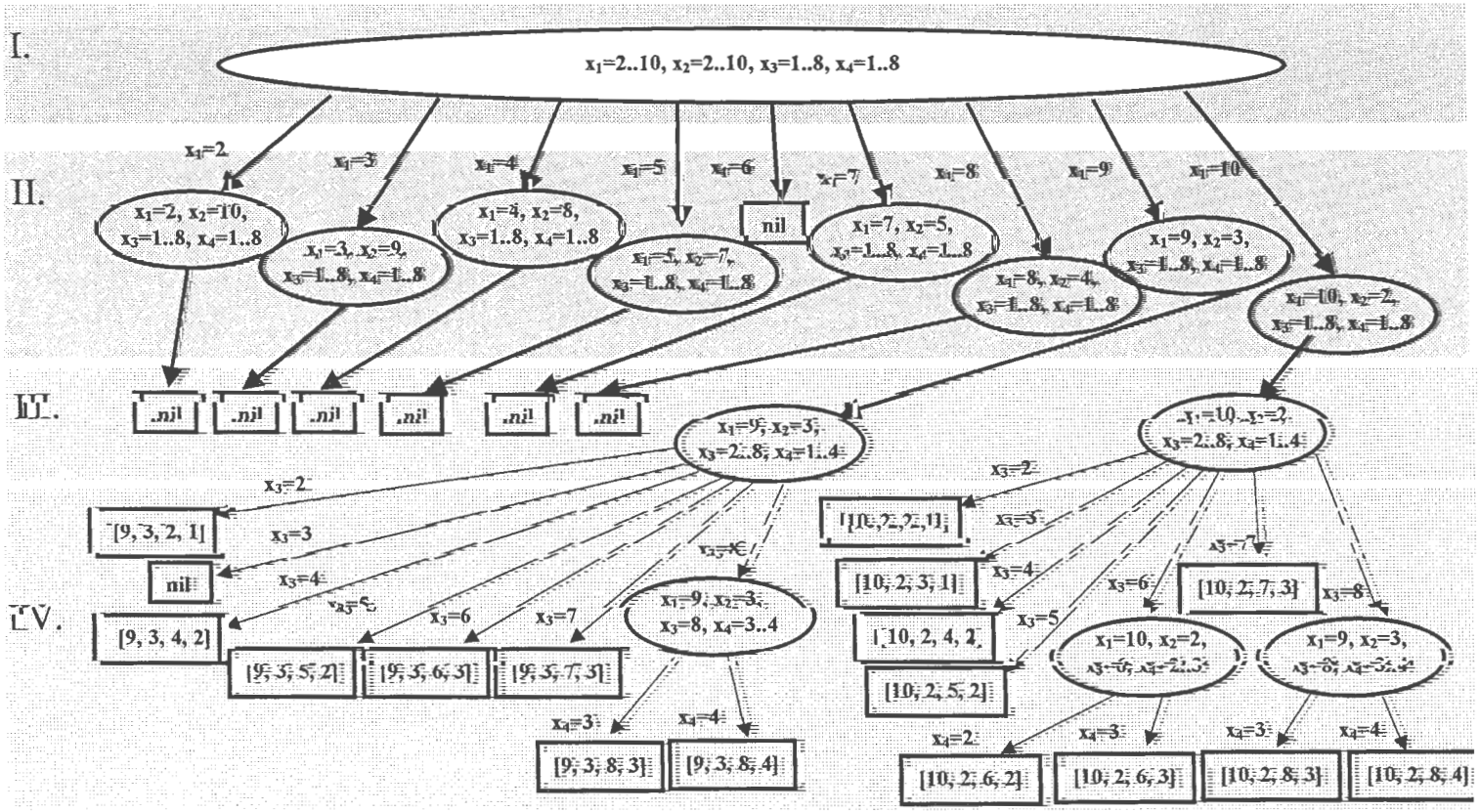
Rys. 17. Uproszczone schematy implementacji wariantów rozwiązania problemu ***PZO***.

Wyniki przeprowadzonych eksperymentów zestawione zostały w Tabeli 2.

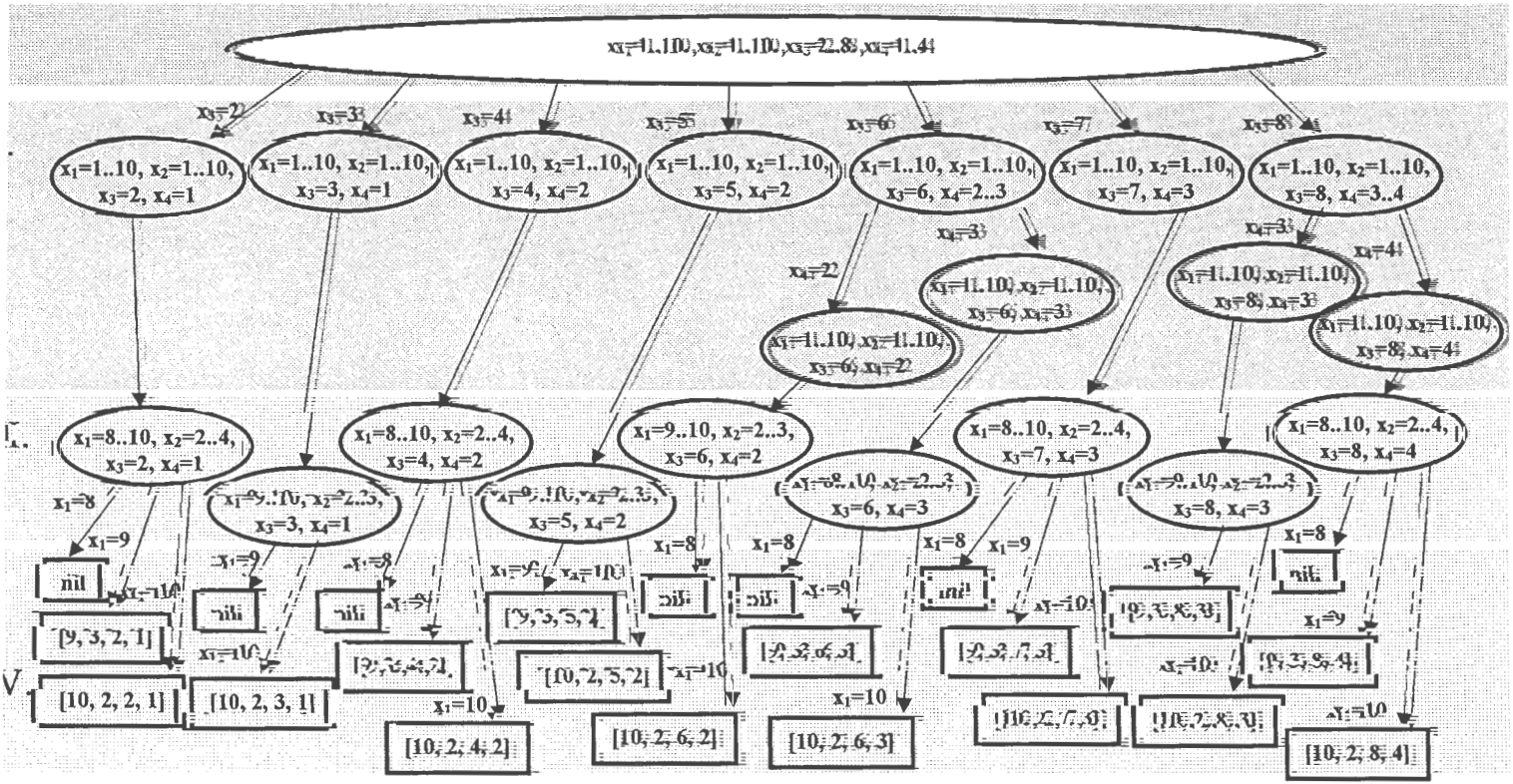
Tabela 2. Zestawienie wyników eksperymentów

Wariant	Liczba rozwiązań dopuszczalnych	Liczba usunięć niezgodności i podstawień	Czas obliczeń
1	16	37	pon. 10 ms
2	16	40	10 ms
3	16	29	pon. 10 ms

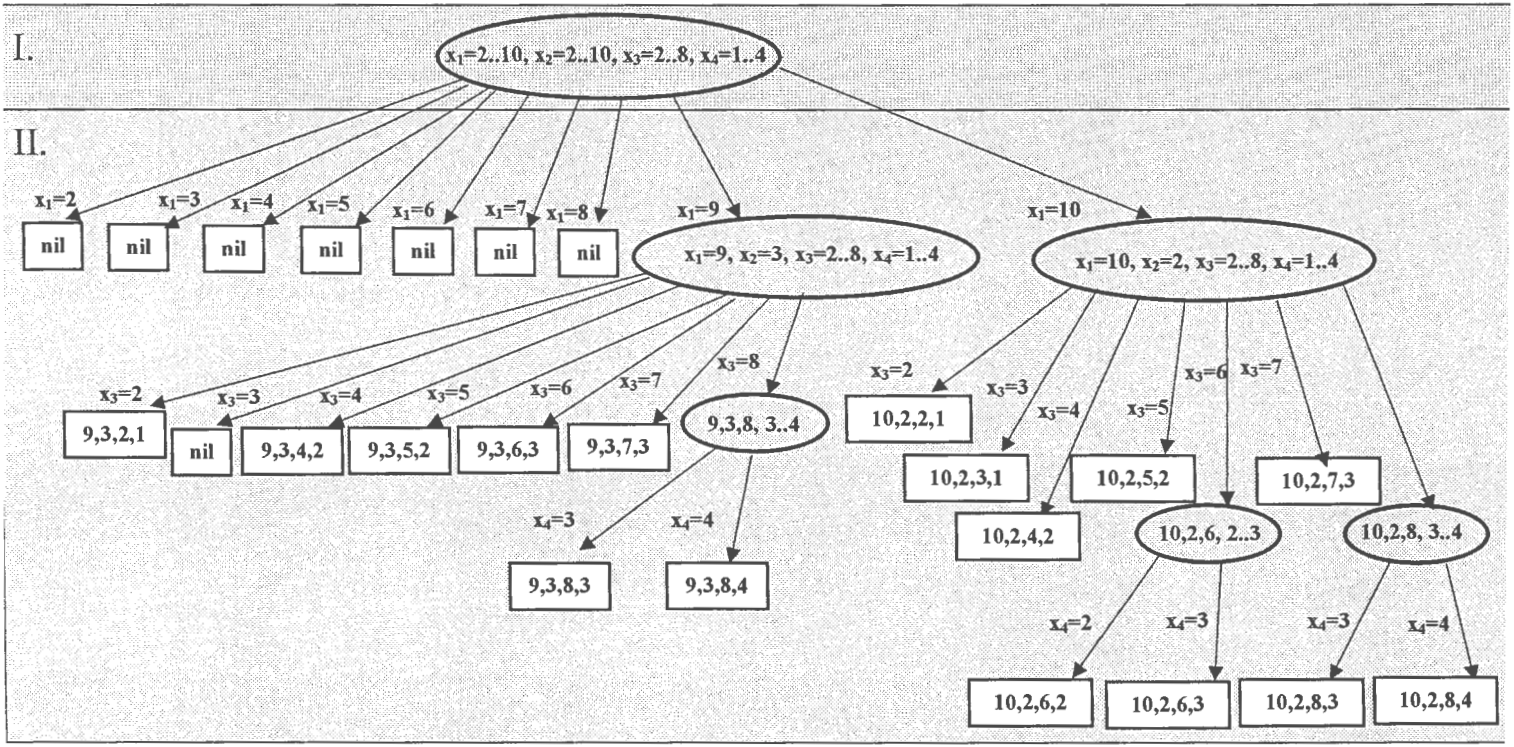
Rysunki 18, 19 i 20 przedstawiają drzewa przeszukiwania według poszczególnych wariantów, odpowiednio wariantu 1, 2 i 3.



Rys. 18. Schemat przeszukiwania przestrzeni rozwiązań wg wariantu 1.

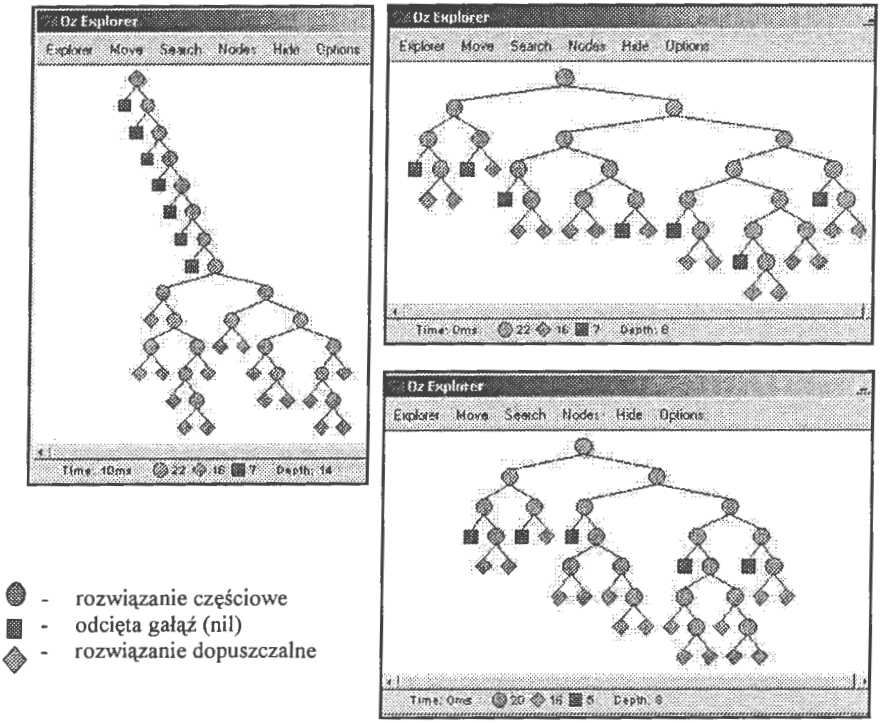


Rys. 19. Schemat przeszukiwania przestrzeni rozwiązań wg wariantu 2.



Rys. 20. Schemat przeszukiwania przestrzeni rozwiązań wg wariantu 3.

Zrzuty z ekranu, odpowiadające poszczególnym wariantom, przedstawia Rys 21.



Rys. 21. Graficzna reprezentacja drzewa przeszukiwania dla poszczególnych wariantów. Rozwiązania uzyskane w języku *Oz* systemu *Mozart*.

W tabeli 3 zebrane zostały liczby dopuszczalnych i potencjalnych rozwiązań uzyskane w poszczególnych wariantach 1-3, w kolejnych etapach (oznaczonych liczbami rzymskimi od I do IV (patrz Rys. 18 – Rys.20)

Tabela 3. Zawężanie potencjalnego obszaru rozwiązań.

Wariant	Początkowa liczba potencjalnych rozwiązań	Liczba potencjalnych rozwiązań			Liczba dopuszczalnych rozwiązań po etapie IV
		po etapie I	po etapie II	po etapie III	
1	6400	656	512	56	16
2	6400	2800	900	61	16
3	6400	2268	16*	-	-

* - Liczba dopuszczalnych rozwiązań.

Jak widać, kolejność w jakiej rozwiązywane są poszczególne podproblemy ma istotny wpływ na „efektywność" zawężania obszarów potencjalnych rozwiązań. Potwierdzony został również fakt, że najefektywniejszym sposobem programowania jest poszukiwanie takich sposobów formułowania problemu *PZO* (a dokładniej zbioru ograniczeń), które umożliwia jego implementację bez konieczności dekomponowania na podproblemy.

4.2 Programowanie

Efektywność programów implementowanych w językach *CP/CLP* zależy od struktury danych (tzn. specyfikacji problemów i możliwości implementacji ograniczeń oferowanych w języku programowania) oraz zastosowanej strategii przeszukiwania. Związane z tym możliwości sprowadzają się, w zasadzie, do dwóch opcji: bądź to do „dopasowania" struktury programu (uwzględniając w tym zarówno strategie przeszukiwania, jak i dostępny język programowania) do zadanej specyfikacji problemu (ograniczeń i zakresu dziedzin zmiennych decyzyjnych), bądź też do takiej specyfikacji problemu, która najlepiej wpisuje się w możliwości dostępnego oprogramowania (w tym również dostępnych strategii przeszukiwania).

Zgodnie z pierwszym podejściem, poszukując struktury programu rokującej efektywne rozwiązanie *PZO* należy rozważyć wybór:

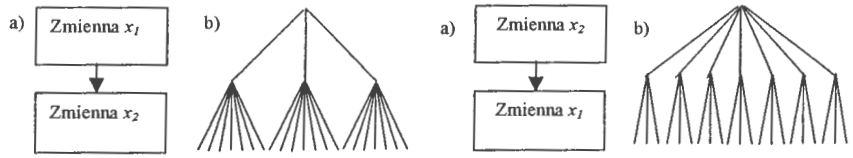
- sposobu i kolejności wywołań procedur propagacji ograniczeń oraz procedur podstawiania,
- strategię podstawiania zmiennych, a w szczególności kolejność podstawiania zmiennych,
- dokładności specyfikowanych wartości zmiennych decyzyjnych.

Sposób i kolejność wywołań procedur propagacji ograniczeń oraz procedur podstawiania ma istotne znaczenie, gdy poszukiwanie rozwiązania odbywa się etapami – problem wyjściowy dekomponuje się na szereg podproblemów (por. rozdział 4.1).

Z kolei, dokonując wyboru strategii podstawiania zmiennych, a w szczególności kolejności podstawiania zmiennych warto przeanalizować np. czy nie korzystniej jest lepiej podstawiać zmienne decyzyjne poczynając od górnych, czy też może poczynając od dolnych wartości ich dziedzin. W językach wykorzystujących metody *CP* jako podstawowa jest wykorzystywana strategia *first fail*, implementująca podstawianie wartości zmiennych rozpoczynające się od dolnych granic dziedzin. Rozważając problem dystrybucji towarów zastosowanie strategii „od dołu" będzie skutkowało wieloma partiami o małej liczności (w skrajnym przypadku w pierwszym kroku będą to partie o liczności równej 1). Łatwo zauważyć, że liczba partii wpływa bezpośrednio na łączną liczbę zadań realizowanych w systemie. Istnieje, zatem duże prawdopodobieństwo, że dla dużej liczby partii o małej liczności, nie zostaną znalezione rozwiązania dopuszczalne. W innych sytuacjach może się okazać, że to właśnie podstawianie wartości zmiennych poczynając „od góry" powoduje wydłużenie czasu obliczeń.

Warto również zauważyć, że odpowiednio dobrana strategia podstawiania wartości zmiennych decyzyjnych pozwala szybko wyszukiwać „racjonalne" rozwiązania dopuszczalne. Przykładowo, łatwo można uniknąć sytuacji, w której znalezione rozwiązanie dopuszczalne przewiduje, np. przewiezienie 30 palet towaru, na 30 TIR-ach (po jednej palecie na ciężarówce), gdyż akurat tyle ciężarówek jest dostępnych

W wielu przypadkach czas poszukiwania rozwiązania zależy od kolejności w jakiej podstawiane są zmienne decyzyjne. Ma to bezpośredni związek ze strukturą drzewa przeszukiwania potencjalnych podstawień. Oznacza to, że programując powinno się dążyć do takiej struktury drzewa, w której zmienne decyzyjne o największych rozmiarach dziedzin są podstawiane jako ostatnie. Rozwiązanie takie ogranicza liczbę nawrotów (Rys. 22).



Rys. 22. Wpływ kolejności podstawiania wartości zmiennych decyzyjnych (a), na kształt drzewa przeszukiwania (b)

Także dokładność („ziarnistość") deklarowanych wartości zmiennych decyzyjnych ma istotny wpływ, determinując wielkości dziedzin zmiennych decyzyjnych, na czas obliczeń. Przykładowo, rozważmy zlecenie transportowe związane z przewiezieniem 100 jednostek towaru. Niech $P = 50$ oznacza dopuszczalną ładowność środka transportu. Konieczne jest, zatem wprowadzenie ograniczenia $P \leq 50$. Zapis taki dopuszcza jednak nieracjonalne, aczkolwiek dopuszczalne rozwiązania, np. 4 partie transportowe (2x49 sztuk i 2x1 sztuka towaru). Celem urealnienia rozwiązań dopuszczalnych warto odrzucić część z nich dopisując dodatkowe ograniczenie, np.: $P \geq 15$ (w najgorszym przypadku oznaczać to będzie 6 partii transportowych z co najmniej 30 procentowym wykorzystaniem środka transportu).

Inną możliwość stwarza podanie *explicite* wszystkich wartości, jakie dane zmienna decyzyjna może przyjąć. Przykładów takiego podejścia dostarcza modelowanie systemów produkcyjnych, w których partie produkcyjne są znacznie większe niż transportowe. Przyjmując, że pojedyncza partia transportowa wynosi $P_T = 5$ sztuk, partia produkcyjna powinna przyjmować wartości P_P=0, 5, 10, 15...itd. Jeśli ograniczeniem górnym partii produkcyjnej jest 20 sztuk towaru, to tak zdefiniowana dziedzina będzie zawierała 5 elementów, a nie 21, co skutkuje w skróceniu czasu obliczeń.

Alternatywne, w stosunku do wyżej przedstawionych możliwości, podejście do zwiększenia efektywności programowania ***PC***, akcentuje potrzebę odpowiedniej specyfikacji ***PZO***. Oznacza to dążenie do takiej implementacji problemu ***PZO***, która wymagałaby jego

minimalnej dekompozycji (por. rozdział 4.1), a w konsekwencji poszukiwanie takiej struktury programu, w której liczba etapów poszukiwania (podproblemów) będzie minimalna, a kolejność ich rozwiązywania pozwoli na optymalne wykorzystanie dostępnych procedur propagacji ograniczeń (unikanie nawrotów).

Z uwagi na fakt, że zmienne decyzyjne reprezentowane są, na ogół, w postaci wektorowej, dobrą praktyką programistyczną jest takie ustawianie zmiennych w tym wektorze aby wartość i-tej zmiennej miała bezpośredni wpływ na i+1zmienną decyzyjną.

Istotną rolę odgrywa, również analiza specyfikowanych ograniczeń. Od ich sformułowania zależy, bowiem często efektywność programowanego w ten sposób mechanizmu selekcji. Celem ilustracji rozważmy programową implementację ograniczenia: $x_i \geq x_{i+1}$.

```
X={List.make 6};                % deklaracja 6 elementowej zmiennej decyzyjnej
X:::0#5                         % dziedziny elementów od 0 do 5
{For 1 5 1
    proc{$I}
        X(I)>=X(I+1)            % zależności pomiędzy zmiennymi decyzyjnymi
    End}
{FD.distinct X}                 % każdy element o innej wartości
```

Łatwo zauważyć, że tak sformułowane ograniczenie nie zmniejsza dziedzin zmiennych decyzyjnych x_i=[0..5], i={1,2,3,4,5}.

Analiza struktury programu wymaga, na ogół, opracowania i przeliczenia kilku do kilkunastu wersji alternatywnych. Postępowanie takie pozwala ocenić wpływ arbitralnie wyodrębnianych (np. wynikających z doświadczenia programisty) obszarów dziedzin poszczególnych zmiennych na pozostałe, jak również ocenić wielkość nakładów obliczeniowych ponoszonych na znalezienie rozwiązania, a także jego jakość.

Przedstawione uwagi podkreślają rolę programisty, od którego doświadczenia i umiejętności zależy efektywność programu finalnego. Szybkość obliczeń zależy, bowiem od umiejętnego (uwzględniającego m.in. ograniczenia implementacyjne dostępnego języka *CP*) sformułowania problemu *PZO*, umiejętnego kształtowania dziedzin zmiennych decyzyjnych, a także prowadzonych analiz statystycznych (uwzględniających m.in. dobór strategii podstawiania i propagacji ograniczeń).

4.3 Wspomaganie decyzji

Większość decyzji, podejmowanych w praktyce przemysłowej, dotyczy sytuacji związanych z bilansowaniem się potrzeb i możliwości. Przykładową potrzebą może być tu podjęcie się przez przedsiębiorstwo realizacji nowego zlecenia produkcyjnego, a gwarantującymi jego wykonanie (lub nie) możliwościami (odpowiednimi zdolnościami produkcyjnymi tego przedsiębiorstwa). Dla ustalenia uwagi przyjmijmy, że zlecenie scharakteryzowane jest wielkością produkcji, terminem jej ukończenia oraz ceną zakupu, podczas gdy zdolności produkcyjne przedsiębiorstwa determinują: czasowe ograniczenia dostępność do poszczególnych stanowisk produkcyjnych, koszty wykorzystania tych stanowisk, dostępne technologie, itp. Łatwo zauważyć, że odpowiedzi na pytania typu: Jaka organizacja przepływu produkcji pozwoli zmaksymalizować wskaźnik wykorzystania zasobów produkcyjnych? Jaka kolejność obsługi zleceń zapewni ich terminową realizację? nabierają sensu, dopiero w momencie gdy wiadomo, że istnieje niepusty zbiór rozwiązań dopuszczalnych.

Spośród innych, często spotykanych w praktyce, problemów *PZO* warto tutaj wspomnieć o problemie przydziału doków pasażerskich dla lądujących samolotów, przydziału odcinków nabrzeży do cumowania statków w portach przeładunkowych, przydziału personelu do

dyżurów w szpitalach, przydziału załóg w lotach pasażerskich, itp. Wspólną cechą tych problemów jest ich *NP*-trudny charakter oraz konieczność szybkiego, w trybie na bieżąco, podejmowania decyzji. Oznacza to potrzebę poszukiwania rozwiązań kompromisowych, tzn. takich sformułowań problemów decyzyjnych i takich technik i metod ich rozwiązywania, które spełniałyby zadane ograniczenia czasowe.

Oprócz ograniczeń czasowych, istotną rolę odgrywają również ograniczenia kosztowe związane z wypracowaniem decyzji. Na ograniczenia te składają się głównie koszty zakupu odpowiedniej technologii *IT* (oprogramowania) oraz koszty związane z zatrudnieniem odpowiedniego personelu. Ostatni z wymienionych wyżej czynników odgrywa szczególnie ważną rolę w problemach optymalizacyjnych, gdzie kwalifikacje i doświadczenie programisty decyduje o jakości rozwiązania. Przykładem odpowiednich problemów optymalizacyjnych są tutaj problemy szeregowania zadań (w szczególności, w produkcji zorientowanej na klienta *(ang. on-demand manufacturing)*) oraz problemy trasowania instalacji (np. okablowania, przewodów wodno-kanalizacyjnych, itp.) w budynkach i na statkach, a także rozmieszczania stacji przekaźnikowych telefonii komórkowej.

Dostępne w tym zakresie systemy wsparcia decyzji bądź to powstawały na bazie ogólnie dostępnych metod badań operacyjnych (stanowiąc ich zadaniowo zorientowane implementacje, np. *Matlab*), bądź też stanowiły ich odpowiednie rozszerzenia (w ramach specjalnych programów badawczych czy komercyjnych). Implementacje technik programowania *CP/CLP* (oprócz wcześniej wspomnianych języków programowania) coraz częściej odnaleźć można również w dedykowanych systemach wsparcia decyzji. Przykładem tego są systemy *ProALPHA* [10] i system *OptiTrans* [16], implementujące język programowania *CP/CLP Ilog*. Pierwszy z nich (w swojej aktualnej wersji) przeznaczony jest do wspomagania *MŚP* w zakresie controllingu, drugi natomiast wspiera procesy obsługi sieci

dystrybucji towarów płynnych. Warto tutaj dodać, że *Ilog* łączy możliwości programowania *CP/CLP* z klasycznymi metodami optymalizacji numerycznej.

Z perspektywy ***MŚP*** oznacza to możliwość budowy tanich, łatwych w obsłudze, a zwłaszcza dedykowanych dla potrzeb przedsiębiorstwa systemów wspomagania decyzji. Możliwości te wiążą się z budową interfejsu użytkownika, interfejsu pozwalającego na obsługę wybranych, standardowych problemów ***PZO***, na bazie ogólnie dostępnych języków programowania (jak np. język *Oz* systemu *Mozart*). W zależności od potrzeb użytkownika, rozwiązania tego typu mogą obejmować (uwzględniające specyfikę przedsiębiorstwa) zagadnienia planowania produkcji, harmonogramowania systemów transportowych, obsługi magazynów, itp. Warto podkreślić, że ze względu na właściwości programowania ***CP***, można pominąć ograniczenia związane z poziomem złożoności rozważanego problemu decyzyjnego. W szczególności oznacza to, że mogą być rozważane zagadnienia integrujące, zwyczajowo oddzielnie rozwiązywane, problemy marszrutowania, porcjowania i harmonogramowania.

Istota, przedstawionej koncepcji budowy systemów wspierających ***MŚP*** sprowadza się do specyfikacji problemów użytkownika, wyboru języka programowania ***CP*** oraz opracowania i implementacji odpowiednich strategii przeszukiwania. Warto zauważyć, że jej realizacja wymaga rozwiązania ciągle otwartego problemu programowania zapewniającego zadany poziom efektywności *(ang. performance debugging)* ciągle należy do problemów otwartych. Dotychczasowe próby jego rozwiązania bazują na analizie statystycznej wybranych strategii przeszukiwania, w pewnych klasach problemów ***PZO*** [8]. Alternatywnym podejściem wydaje się być również rozwiązanie zmierzające do podania warunków wystarczających (wiążących parametry ***PZO***, ograniczenia implementacyjne języków ***CP*** oraz strategie przeszukiwania), spełnienie których gwarantowałoby pożądaną jakość programu.

5 Zakończenie

Do podstawowych zalet programowania ograniczeń należy zaliczyć możliwość: integracji różnych (zwyczajowo oddzielnie rozwiązywanych) problemów cząstkowych oraz koncentracji na specyfikacji problemu (w miejsce akcentowania sposobu rozwiązania problemu). Pomimo wielu, niewątpliwych sukcesów, związanych z jego praktycznymi implementacjami, istnie wiele, ciągle otwartych, problemów. Należą do nich problemy doboru technik rozwiązywania dla poszczególnych ***PZO***, oceny nakładów (zwykle czasowych) związanych z wyznaczeniem rozwiązania optymalnego, zapewnienia pożądanego poziomu efektywności programu, itp.

Rosnąca konkurencja na rynku ***MŚP*** rodzi zapotrzebowanie na szybkie, tanie i łatwe w obsłudze systemy wsparcia decyzyjnego. Wymienione wyżej zalety programowania ograniczeń dostarczają atrakcyjnej alternatywy wobec aktualnie dostępnych rozwiązań systemów klasy ***ERP***. W szczególności dotyczyć to może budowy zadaniowo zorientowanych interfejsów, umożliwiających użytkownikowi szybkie podejmowanie standardowych decyzji.

Literatura

[1] Adamczewski A., Zintegrowane Systemy Informatyczne w praktyce, Warszawa 2001

[2] Bartak R., On-line guide to constraint programming, Prague, 1998, http://kti.mff.cuni.cz/~bartak/constraints/

[3] Bartak R., Incomplete dept-first search techniques: A short survey. Proc. of the 6th Workshop on Constraint programming for Desision and Control, Ed. J. Figwer, June 29, 2004, Silesian University of Technology, Gliwice, pp.7-14.

[4] Bartak R., Constraint programming: In pursuit of the holy grail. http://kti.mff.cuni.cz/~bartak/constraints/

[5] Banaszak Z., Józefowska J. Red. Project-driven manufacturing. WNT, Warszawa 2003.

[6] Banaszak Z., Pisz I., Project–driven production flow management. In. Project Driven Manufacturing. Banaszak Z. J. Józefowska Eds, WNT, Warszawa, 2003, pp. 53-71.

[7] Banaszak Z., M. Zaremba,CLP-based project-driven manufacturing. Prepr. Of the 7th IFAC Symposium on Cost Oriented Automation, June 6-9, 2004, Gatineau, Quebec, Canada, pp.269-274.

[8] Borowiecki T., Banaszak Z., Stryjski R., An approach for CSP/COP modelling of decision making procedures. Proc. of the 6th Workshop on Constraint programming for Desision and Control, Ed. J. Figwer, June 29, 2004, Silesian University of Technology, Gliwice, pp.21-31.

[9] Bzdyra K., Z. Banaszak, Zastosowanie technik programowania z ograniczeniami w zintegrowanym planowaniu przepływu produkcji. Polioptymalizacja i komputerowe wspomaganie projektowania. Tom III, Red. W. Tarnowski, T. Kiczkowiak, WNT Warszawa, 2004, pp.25-34

[10] Kluge P.D., Controlling w małych i średnich przedsiębiorstwach, Kluge P.D., P. Kuźdowicz (red.) Controlling wspomagany komputerowo w małych i średnich przedsiębiorstwach; Zielona Góra, 2004

[11] Nowosielski S. Narzędzia informatyzacji controllingu w małych i średnich przedsiębiorstwach. Analiza wyników badań, Zielona Góra , Kluge P.D., P. Kuźdowicz (red.) Controlling wspomagany komputerowo w małych i średnich przedsiębiorstwach; 2004,

[12] Mejssner M., Od stanowiska do sieci. Finansista Nr. 9, 2003, pp.66-67.

[13] Patalas J., Kłos S., Banaszak Z.,Rola centrum transferu technologii w przygotowaniu projektów informatycznych wdrażania systemów komputerowo zintegrowanego zarządzania MSP. (w druku)

[14] Sitek P., Wikarek J., Banaszak Z., Wykorzystanie elementów aktywnych współczesnych baz danych do implementacji pewnej metody planowania produkcji, Zeszyty Naukowe Wydz. Mechanicznego PK, nr 35, Wyd. Uczelniane PK, Koszalin 2004

[15] Zielińska D., Zintegrowane wspomaganie zarządzania przedsiębiorstwem: Przyszłość systemów ERP, <http://www.bankier.pl/

[16] www:at-consulting.pl