1.12 — metody numeryczne
i komputerowe

Stanisław Jóźwiak

SYSTEMY WIEDZY W OPTYMALIZACJI

3/1990

WARSZAWA 1990

ISSN 0208-5658

Praca wpłynęła do Redakcji dnia 31 sierpnia 1990 r.

IPPT PAN

Na prawach rękopisu

Instytut Podstawowych Problemów Techniki PAN
Nakład 140 egz. Ark.wyd. 2,05 Ark.druk. 2,75
Oddana do drukarni w lutym 1990 r.
Nr zamówienia 51/90

Warszawska Drukarnia Naukowa, Warszawa
ul. Śniadeckich 8

Stanisław Jóźwiak
Pracownia Systemów Adaptacyjnych

SYSTEMY WIEDZY W OPTYMALIZACJI

Streszczenie

W pracy dokonano przeglądu literatury dotyczącej wykorzystania metod sztucznej inteligencji w optymalizacji, ze szczególnym uwzględnieniem zastosowania w projektowaniu konstrukcji. Przegląd poprzedzono krótkim omówieniem podstawowych pojęć i technik z dziedziny sztucznej inteligencji, a także ich praktycznego wykorzystania przy tworzeniu programów komputerowych. Przedstawiono sposoby poprawy efektywności programów optymalizacyjnych dzięki wykorzystaniu metod sztucznej inteligencji w podstawowych etapach rozwiązywania zadania projektowego. Obejmują one: sformułowanie modelu, wybór metody rozwiązywania zadania programowania matematycznego i dobór parametrów, diagnostykę i ocenę rozwiązania. Podano przykłady zrealizowanych programów i systemów komputerowych przeznaczonych do projektowania elementów i układów konstrukcyjnych i mechanicznych.

1. WSTĘP

Metody optymalizacji zostały uznane za użyteczne narzędzie w projektowaniu konstrukcji już w latach 50-tych. Pomimo, że od tego czasu uczyniono znaczny postęp zarówno w dziedzinie samej optymalizacji jak i w sposobie rozwiązywania zadań projektowych a także, a może przede wszystkim, w zakresie technicznego wyposażenia (komputery), metody optymalizacji jako sposób poprawy efektu pracy inżyniera są w dalszym ciągu niedoceniane i niewykorzystywane. Porównajmy, na przykład, postęp jaki dokonał się w ostatnich latach w zakresie upowszechnienia metody elementów skończonych (MES). Jako punkt

odniesienia mozna przyjąć termin Drugiej Konferencji ASCE nt. Technik Komputerowych (1960, Pittsburg[46]), na której prezentowano MES i metody optymalizacji. Od tego czasu MES stała się jedną z podstawowych technik w zakresie mechaniki stosowanej, podczas gdy optymalizacja w dalszym ciagu walczy o należyte miejsce w praktyce inżynierskiej.

Wśród przyczyn tego stanu, jedną z najistotniejszych jest fakt, ze efektywne stosowanie metod optymalizacji dla rozwiązywania praktycznych zadań inżynierskich wymaga równoczesnej bardzo dobrej znajomości technik optymalizacji oraz pełnego zrozumienia istoty rozwiązywanego problemu projektowego. Konieczność korzystania z usług wysokiej klasy specjalistów (ekspertów) jest istotną przeszkodą w praktycznym stosowaniu metod optymalizacji jednocześnie jednak sprawia, ze dziedzina ta staje się idealnym miejscem dla zastosowania systemów wiedzy.

2. WPROWADZENIE DO ZAGADNIEŃ SZTUCZNEJ INTELIGENCJI

2.1 Wybrane pojęcia z zakresu sztucznej inteligencji

Ponizej dokonano krótkiego omówienia wybranych zagadnień związanych z wykorzystaniem technik sztucznej inteligencji (Sz. I.). Należy przy tym zauważyć, ze obecnie brak jednoznacznej definicji pojęcia określanego mianem sztuczna inteligencja. Nawet w pracach uważanych za podstawowe i wiodące w dziedzinie jak np. monografia Winstona [45], używa się określeń opisowych starając się raczej przekazać czytelnikowi ogólne wyobrażenie autora niż podać precyzyjne określenia. Zazwyczaj przyjmuje się, ze Sz. I. to dział informatyki, którego celem jest opracowanie technik i metod ułatwiających korzystanie z komputerów przez tworzenie programów, w których sposób przetwarzania informacji jest zbliżony do sposobu rozwiązywania problemów stosowanego przez człowieka. Obszary zastosowania Sz.I. obejmują, między innymi: przetwarzanie języka naturalnego, uczenie maszynowe (machine learning), automatyczne programowanie, inżynierię wiedzy, zagadnienia robotyki. Wśród najważniejszych cech programu inteligentnego, w porównaniu z

programem tradycyjnym, wymienia się: mozliwość przetwarzania danych symbolicznych, deklaratywny zapis wiedzy, szerokie wykorzystywanie wiedzy heurystycznej, dynamiczny charakter wiedzy wykorzystywanej w programie, mozliwość śledzenia przebiegu procesu decyzyjnego, interaktywny tryb pracy (jako tryb podstawowy). Wskazuje się takze, ze program inteligentny umozliwia rozwiązanie pewnej klasy zagadnień należących do określonej dziedziny, w tym takze takich, które nie musiały być znane układającemu program. Gwałtowny rozwój Sz. I. rozpoczął się pod koniec lat 60-tych, kiedy to nastąpiła generalna zmiana w podejściu do zagadnienia [11]. W początkowym okresie, podejmowano prace, których celem miało być wyspecyfikowanie elementów stanowiących o inteligencji, a następnie opracowanie inteligentnego systemu ogólnego przeznaczenia (general-purpose intelligent system). Nacisk kładziono na stworzenie efektywego aparatu wnioskującego, zaniedbując problematykę związaną z akwizycją i sposobami wykorzystania wiedzy przez system. Prace te nie doprowadziły do oczekiwanego rezultatu i zmusiły do zmiany kierunku badań. Przedmiotem zainteresowania stała się wiedza: określenie co stanowi wiedzę, metody akw zycji i reprezentowania w sposób dogodny dla systemu cyfrowego, a takze przetwarzanie wiedzy. Problemami tymi zajmuje się dział Sz.I. zwany inzynieria wiedzy. Dla celów Sz. I. przyjmuje się, ze wiedza (z określonej dziedziny) obejmuje opisy obiektów, relacje i działania (operacje) jakie mozna wykonywać (na obiektach i relacjach) w ramach opisywanej dziedziny. Opisy zawieraja informacje o obiektach, umozliwiają ich identyfikację i rozróżnianie. Relacje określają zaleznosci pomiędzy obiektami. Działania umozliwiają łączenie i porównywanie obiektów dla osiągnięcia celu jakim jest rozwiązanie problemu. Zgodnie z terminologią stosowaną w inzynierii wiedzy opisy obiektów wraz ze znanymi relacjami tworzą tzw. bazę wiedzy zaś działania wykonywane na zawartości bazy są w systemach obliczeniowych reprezentowane jako odpowiednie moduły lub programy.

Akwizycja wiedzy obejmuje wyspecyfikowanie odpowiedniej wiedzy z danej dziedziny, stanowiącej merytoryczną zawartość bazy wiedzy systemu. Wymaga to, zazwyczaj, udziału jednej lub kilku osób (ekspertów) reprezentujących dobrą znajomość problematyki, a takze specjalisty z zakresu inzynierii wiedzy, który potrafi przekształcić wiedzę eksperta

do postaci zgodnej z przyjętym formalizmem reprezentowania wiedzy.

Inzynieria wiedzy wyróżnia kilka sposobów reprezentowania wiedzy, z których najszersze zastosowanie znalazły: systemy regułowe, sieci semantyczne i ramki. Z innych formalizmów wymienić można, między innymi, grafy i rachunek predykatów.

Systemy regułowe

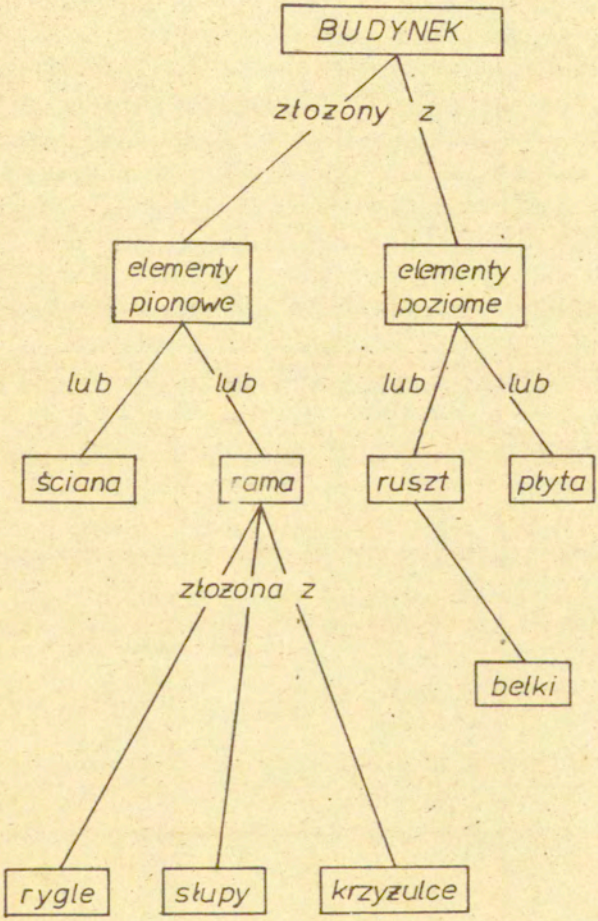
Systemy regułowe (rule-based systems) są obecnie uważane za najbardziej efektywny sposób reprezentowania wiedzy [18]. Podstawowe elementy systemu regułowego to baza wiedzy i mechanizm wnioskujący. Baza wiedzy zawiera fakty oraz reguły o postaci:

IF <*warunek*> THEN <*akcja*>.

W trakcie pracy programu, w przypadku gdy <*warunek*> jest spełniony wykonywana jest <*akcja*>, przy czym element oznaczony symbolicznie jako <*akcja*> może oznaczać czynność jaka ma być wykonana lub konkluzję jaka jest formułowana w przypadku gdy zachodzi <*warunek*>. Właściwe operowanie regułami i faktami wymaga opracowania odpowiedniej strategii postępowania. Wyróżnia się dwie podstawowe strategie postępowania (wnioskowania): wnioskowanie wprzód (forward chaining) oraz wnioskowanie wstecz (backward chaining). Wnioskowanie wprzód stosowane jest w przypadkach gdy w momencie rozpoczęcia analizy, dany jest zestaw faktów opisujących rozważane zagadnienie. W trakcie rozwiązywania wykonywane są kolejno reguły, w których <*warunek*> jest zgodny z danymi faktami. Proces trwa aż do osiągnięcia poszukiwanego celu (<*akcja*> zgodna z przyjętym celem) lub stwierdzenia braku rozwiązania. Wnioskowanie wstecz polega na wskazaniu pewnych hipotez, które traktowane są następnie jako prawe strony <*akcje*> w regułach systemu. W tym przypadku dobiera się regułę, w której jako <*akcja*> występuje sprawdzana hipoteza zaś <*warunek*> stanowi poszukiwane rozwiązanie.

Sieci semantyczne

Sieć semantyczna składa się z węzłów reprezentujących obiekty i gałęzi reprezentujących powiązania pomiędzy węzłami. Przykład sieci semantycznej reprezentującej wiedzę o konstrukcji inżynierskiej przedstawiono na rysunku 2.1.



Rys 2.1 Siec semantyczna

Ramki

Jest to technika reprezentowania wiedzy [10] o obiekcie (lub klasie obiektów), w której rezerwuje się miejsce (slots) na zapis wszystkich informacji o obiekcie. Mogą to być wartości liczbowe, opisowe, informacje o połączeniach z innymi ramkami lub odwołania do procedur, które umozliwią obliczenie pewnych wartości w zaleznosci od określonych warunków. Wiedza jest reprezentowana w sposób wysoce strukturalizowany. Koncepcja ramki jest zgodna z intuicyjnie zrozumiałą zasadą, ze zbierając wiedzę o obiekcie czy zjawisku wykorzystuje się istniejące juz szablony-ramy opracowane dla podobnych zagadnień. Wypełnia się tylko odpowiednie przegródki konkretnymi danymi o właściwościach, powiązaniach czy czynnościach. Czasami tez odpowiednie miejsce pozostawia się niewypełnione zakładając, ze informacje zostaną uzupełnione w terminie późniejszym lub tez przyjmuje się, ze decyzję trzeba bedzie podjąć w oparciu wyłącznie o dostępną wiedzę. Przykład reprezentowania wiedzy dotyczącej elementu konstrukcyjnego (belki) przedstawiono na rysunku 2.2.

nazwa: belka
slots:
rodzaj
wartosc: stalowa_typu_I
rozpietosc
wartosc: 6.0
jednostki: m
obciazenie
wartosc: 60
jednostki: kN/m
wytrzymalosc
wyrazenie: q*l*l*D/(16*J)
ugiecie
wyrazenie: 5*q*l*l*l*l/(384*E*J)

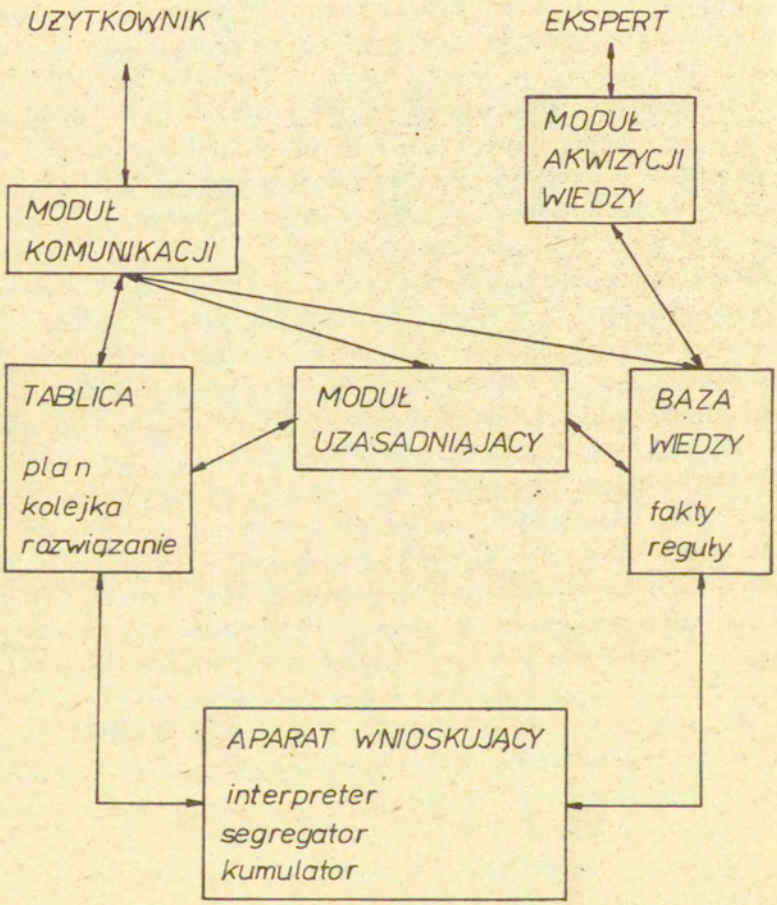
Rys. 2.2 Reprezentacja ramkowa

2.2 Systemy wiedzy

W dalszym ciągu przez system wiedzy (knowledge - based system) będziemy rozumieli program komputerowy przeznaczony do rozwiązywania określonego problemu, naśladujący sposób postępowania człowieka. Podstawowe cechy odróżniające systemy wiedzy od programów tradycyjnych [9] to: jawny sposób reprezentowania wiedzy w formie bazy wiedzy oraz mechanizm wnioskujący umozliwiający automatyczne wnioskowanie w oparciu o wiedzę zgromadzoną w bazie wiedzy i informacje dostarczane przez uzytkownika (w trakcie interakcji). Dla celów niniejszego opracowania rozróżnia się systemy wiedzy, w sensie podanym powyzej, od systemów ekspertowych (expert systems) jako szczególnego rodzaju systemów wiedzy, których mozliwości wynikające z zasobów jakimi dysponuje baza wiedzy powodują, ze program pełnić moze funkcję eksperta w określonej dziedzinie. Aby program mógł być uznany za system ekpertowy musi dysponować dostatecznie bogatą wiedzą, którą określić mozna liczbą reguł zapisanych w bazie wiedzy. Przyjmuje się, ze system ekspertowy mogący mieć praktyczne zastosowanie powinien posiadać kilkaset reguł. (Ocenia się, ze pełna ekspertyza w dziedzinie (pełny system ekspertowy z określonej dziedziny wiedzy) wymaga do 10000 reguł.) Systemy o mniejszej bazie wiedzy mogą być stosowane praktycznie, mozna je takze wykorzystywać do efektownych pokazów, nie powinny jednak nosić miana systemu ekspertowego.

Strukturę idealnego systemu ekspertowego przedstawiono na rysunku 2.3, który zaczerpnięto z klasycznej juz pracy dotyczącej systemów ekspertowych [18]. Ponizej omówione zostaną podstawowe elementy typowego systemu wiedzy.

Komunikacja system-uzytkownik powinna odbywać się "na poziomie uzytkownika" tzn. system powinien być w stanie zrozumieć komendy i informacje podawane przez uzytkownika w sposób naturalny, bez konieczności "transformowania" informacji (przez stosowanie języków problemowo-zorientowanych, słów kluczowych itp.). Moduł komunikacji powinien być więc wyposazony w procesor języka naturalnego oraz umozliwić komunikację przy wykorzystaniu aparatu chrakterystycznego dla danej dziedziny. W przypadku programów inzynierskich szczególne znaczenie będą tu miały moduły graficznego wprowadzania danych i prezentacji wyników typowe dla systemów CAD.



Rys 2 3 Struktura systemu ekspertowego (wg[17])

Kolejnym elementem systemu jest baza wiedzy. Dane zapisane w bazie mozna podzielić na tzw. dane pasywne (context), które odpowiadają faktom dotyczącym aktualnie badanego problemu oraz tzw. dane aktywne, które obejmują obowiązujące reguły. Sposobem wykorzystania reguł zgromadzonych w bazie wiedzy steruje kolejny element systemu tzw. aparat wnioskujący (inference mechanism). W skład aparatu wchodzą: interpreter, segregator i kumulator.

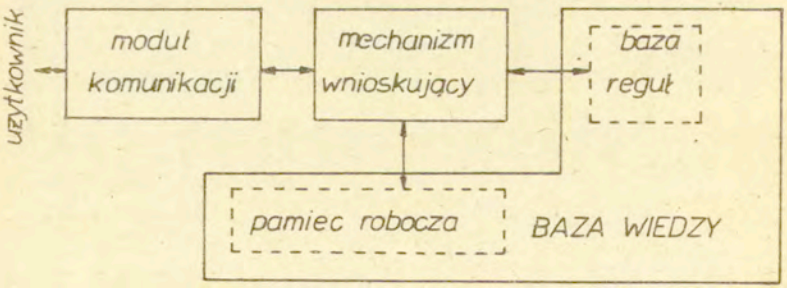
W trakcie pracy system generuje rózne hipotezy, które są nastepnie weryfikowane i modyfikowane w miarę jak postepuje proces rozwiązywania. Pośrednie decyzje i hipotezy są rejestrowane przez wyspecjalizowaną składową systemu tzw. tablicę (blackboard). Części składowe tablicy to: plan, kolejka i roziązanie. Elementy planu określają ogólną strategię przyjętą dla rozwiązywania danego zagadnienia np. w przypadku duzych i złozonych zagadnień rozwiązanie mozna uzyskać tylko na drodze dekompozycji. Wyróżnione podproblemy są rozwiązywane oddzielnie i dopiero zestawienie rezultatów częściowych daje poszukiwane rozwiązanie problemu ogólnego. W takim przypadku plan zawiera strategię według której dokonuje się dekompozycji. Kolejka zawiera listę zadań lub hipotez, które nalezy rozwiązać. Część tablicy zwana rozwiązaniem zawiera częściowe konkluzje i potencjalne hipotezy, a takze informacje o powiązaniach między nimi. Zawyczaj najbardziej obiecujące hipotezy lub konkluzje zebrane w kolejce są wykonywane jako pierwsze. Zawiaduje tym segregator wchodzący w skład aparatu wnioskującego, który ocenia słuszność (validity) konkluzji i hipotez porównując formalną zgodność parametrów hipotez i konkluzji z zawartością bazy wiedzy. Sprawdzenie (wykonanie) hipotezy będzie powodowało wygenerowanie nowej hipotezy lub konkluzji lub tez spowoduje zmiany w elementach tablicy. Kumulator sprawdza zgodność wprowadzonych rozwiązań starając się jak najwcześniej wyeliminować niezgodności i niekonsekwencje.

Istotnym elementem systemu jest moduł uzasadniający (explanation facility), który informuje uzytkownika o merytorycznych przesłankach podejmowanych przez system decyzji, na jakiej podstawie odrzucono inne ewentualne rozwiązania lub dlaczego potrzebne są dodatkowe informacje.

Akwizycja wiedzy jest jednym z kluczowych etapów w budowie systemu wiedzy. Wynika to, między innymi, z faktu, ze w ostatecznym efekcie o jakości systemu decyduje zawartość bazy wiedzy. Spotyka się przy tym opinię, ze to właśnie trudności w zgromadzeniu odpowiedniej wiedzy

stanowią istotną przeszkodę w opracowywaniu systemów ekspertowych. Inzynier wiedzy jest zalezny od eksperta, który sam musi dokonać wyboru, przeprowadzić selekcję i systematyzację posiadanych wiadomości. Czynione są więc próby aby proces ten maksymalnie zobiektywizować i uniezaleznić od czynnika ludzkiego. Polegać to moze np. na oparciu gromadzenia wiedzy na podejściu indukcyjnym, w którym do systemu dostarczane są fakty-przykłady na podstawie których indukowane są prawa rządzące zjawiskiem. Zaletą takiego podejścia jest mozliwość zautomatyzowania procesu gromadzenia wiedzy.

Prowidłowo zbudowany system moze być wykorzystywany w różnych dziedzinach. Zmianie ulec musi jedynie zawartość bazy wiedzy. Praktycznie przy zmianie bazy trzeba zmodyfikować (przeprogramować) przynajmniej aparat wnioskujący, a zwłaszcza segregator. System z pustą bazą wiedzy jest nazywany powłoką systemową (system shell). Obecnie mozna zauważyć próby opracowywania powłok systemowych o przeznaczeniu komercyjnym, które uzytkownik moze wyposazać w bazy wiedzy w zalezności od indywidualnych potrzeb. Takze dostępne obecnie kompilatory ułatwiają tworzenie prostych systemów wiedzy. Na rysunku 2.4 przedstawiono schemat prostego systemu regułowego (opartego o regułową reprezentację wiedzy) napisanego w TURBO PROLOGU [40,41]. Kompilator języka pełni w tym przypadku funkcję mechanizmu wnioskującego, zaś treść programu, zapisana w postaci zbioru reguł, stanowi bazę wiedzy. Zadaniem programisty jest właściwe zorganizawanie reguł, a takze zaprogramowanie interakcji człowiek-komputer [13].



Rys 2.4 Schemat systemu regułowego [41]

3. SYSTEMY WIEDZY W OPTYMALIZACJI

W niniejszym rozdziale dokonano krótkiego przeglądu istniejącego oprogramowania, a takze omówiono niektóre koncepcje dotyczące wykorzystania systemów wiedzy (S.W.) w podstawowych etapach rozwiązywania zadania optymalizacji. Obejmują one:

- sformułowanie modelu,
- wybór metody rozwiązywania zadania programowania matematycznego i dobór parametrów,
- diagnostykę i ocenę rozwiązania.

Omówione zostaną takze możliwości wykorzystania systemów wiedzy na etapie rozwiązywania zadania programowania matematycznego.

3.1 Formułowanie modelu

W tym etapie podejmowane są podstawowe decyzje mające zasadniczy wpływ na dalszy przebieg procesu obliczeniowego optymalizacji; dokonuje się wyboru zmiennych decyzyjnych i parametrów projektowych, określa się postać funkcji celu i ograniczeń. Ze względu na znaczenie tych czynności, w praktyce, nawet w najprostszych zagadnieniach dokonuje się wielokrotnej rewizji modelu zanim uzyskany efekt uznany zostaje za satysfakcjonujący. Opracowanie dobrego modelu wymaga znajomości rozwiązywanego zagadnienia jak również wiadomości z zakresu matematyki pozwalających ocenić własności sformułowanego modelu i jego przydatność w konkretnym przypadku.

Przykładem realizacji komputerowej jest system EMP (Expert System for Mathematical Programming [37]) wspomagający uzytkownika przy opracowywaniu modelu, a takze w wyborze metody rozwiązania i przy określaniu wartości parametrów dla róznych rodzajów zadań programowania matematycznego, między innymi, liniowego, kwadratowego, nieliniowego, minimaksowego, wielokryterialnego. Współpraca z programem odbywa się w trybie konwersacyjnym przy wykorzystaniu hierarchicznie zbudowanych menu. Dzięki wyposazeniu systemu w bogaty zestaw procedur matematycznych dla kazdego zagadnienia optymalizacji można wskazać przynajmniej dwa rózne algorytmy rozwiązujące. Program rejestruje informacje o efektach zastosowania wybranego algorytmu, a w przypadku niepowodzenia wykorzystując odpowiedni zestaw reguł dokonuje analizy przyczyn niepowodzenie (do wykorzystania w następnych realizacjach).

Konwersacyjny sposób wprowadzania informacji jest stosowany w wielu

nowoopracowywanych [7] programach, często będących udoskonalonymi wersjami starszych realizacji (np. MSPN [36]).

Odmienne zastosowanie S.W. do tworzenia modeli optymalizacyjnych jest związane z wykorzystaniem preprocesorów do analizy symbolicznej (symbolic mathematical processors) jak na przykład REDUCE [19] czy MACSYMA [25]. Wykorzystanie, opartego na LISPie, pakietu REDUCE w projektowaniu omówili, między innymi, Li i Papalambros [20]. Preprocesor analizy symbolicznej spełnia, w tym wypadku, rolę podobną do procesorów słów (word processors) przy tworzeniu tekstów. Próbuje się także stosować procesory analizy symbolicznej w połączeniu z innymi programami np. MES. Takie podejście przedstawiono w pracach Hartmanna [15,16]. Proces optymalizacji jest wspomagany przez wykorzystanie symbolicznego różniczkowania macierzy sztywności względem zmiennych decyzyjnych. Program napisano w jednej z wersji PROLOGu.

W literaturze [31] wspomina się także o innych sposobach wykorzystania S.W.. Jedno z proponowanych podejść, oparte na systemie regułowym, polega na wstępnym dobraniu modelu, który jest następnie oceniany ze względu na własności przyszłego rozwiązania i analizowany z punktu widzenia przydatności w rozpatrywanym zagadnieniu. Wykorzystuje się w tym celu bazę wiedzy systemu, bez konieczności jednak uzyskiwania rozwiązania numerycznego.

3.2 Wybór metody i dobór parametrów

Wybór metody rozwiazania zadania optymalizacji jest prawdopodobnie zagadnieniem, któremu poświęcono najwięcej uwagi nawet w najprostszych systemach konwersacyjnych. Jest to spowodowane nie tylko dużą liczbą istniejących technik i odmian ale także tym, że niektóre z nich są wyraźnie lepsze od innych, przynajmniej w zastosowaniu do określonych zadań. Z punktu widzenia S.W. jest to zagadnienie szczególnie interesujące gdyż eksperci z dziedziny nie zawsze są całkowicie zgodni co do wyboru metody. Wynika to nie tyle z różnicy poglądów na temat działania algorytmu, ale raczej z różnicy w interpretacji obserwowanego zachowania programu i odmiennych priorytetów. Zagadnienie to było, między innymi, przedmiotem badań Lootsma [24], który sformułował wymagania dla przyszłego systemu ekspertowego. System expertowy do wyboru techniki optymalizacji powinien koordynować wiedzę o modelu i dostępnych technikach. Wiedza o modelu powinna

umozliwić dokonanie klasyfikacji ze względu na podstawowe własności istotne przy doborze metody (liniowy, kwadratowy, nieliniowy, mozliwości różniczkowania analitycznego etc.). System powinien takze dokonać następujących czynności:

- oszacować "koszt obliczeniowy" potrzebny do wyznaczenia funkcji (celu i ograniczeń) i użyć tej informacji do wstępnego zasugerowania metody i parametrów;
- podpowiedzieć i zaimplementować sposoby skalowania związane z wybraną metodą;
- sprawdzić ograniczenia związane z wybraną metodą;
- sprecyzować preferencje uzytkownika i w oparciu o posiadaną wiedze wskazać metodę;
- sprawdzić stopień wypełnienia macierzy w stosunku do wielkości zadania i zadecydować czy nalezy zastosować specjalną technikę rozwiązywania.

Jak dotychczas są to jedynie sugestie, a główną ideą jest, ze istnieje stosunkowo znaczna liczba parametrów, których ocena moze być zaprogramowana jeśli tylko istnieje (techniczna) mozliwość zautomatyzowania podejmowania decyzji w zalezności od pewnych warunków. Szczególnie wygodna okazuje się w tym wypadku regułowa reprezentacja wiedzy. Przykłady reguł stosowanych dla wyboru algorytmu rozwiązywania zadania optymalizacji podano w pracy [6]:

IF *wszystkie zmienne sa ciagle*
AND *wszystkie ograniczenia sa liniowe*
AND *funkcja celu jest liniowa*
THEN *konkluzja: programowanie liniowe*
AND *wykonaj algorytm LP*

IF *wszystkie zmienne sa ciagle*
AND *wszystkie ograniczenia sa liniowe*
AND *funkcja celu kwadratowa*
THEN *konkluzja: programowanie kwadratowe*
AND *wykonaj algorytn QP*

IF *wszystkie zmienne sa calkowite*
AND *wszystkie ograniczenia sa liniowe*
AND *funkcja celu liniowa*
THEN *konkluzja: programowanie liniowe, calkowitoliczbowe*
AND *wykonaj algorytm ILP*

IF *wszystkie zmienne sa ciagle*
AND *wszystkie ograniczenia sa liniowe*
AND *liczba funkcji celu wieksza od 1*
AND *wszystkie funkcje celu sa liniowe*
THEN *konkluzja: programowanie wielokryterialne, liniowe*
AND *wykonaj algorytm MOLP*

Zblizony sposób postępowaniu zastosowano we wspomnianym juz systemie EMP. Dla rozwiązania ogólnego zadania NLP w zwykłych warunkach, system uzyje metody SQP. W przypadku nie uzyskania pozytywnego rezultatu, zadanie jest automatycznie rozwiązywane przy wykorzystaniu metody "eliptycznej".

Typowym przykładem danych, których znaczenie dla przeciętnego uzytkownika nie zawsze jest całkowicie zrozumiałe są parametry programu. Przyjęcie odpowienich wartości moze mieć istotny wpływ na końcowy wynik działania algorytmu, jednocześnie jednak określenie wielkości kryterium stopu, dokładności spełnienia ograniczeń, adresów zmiennych decyzyjnych, przewidywanej liczby iteracji stwarza projektantowi nie dysponującemu większą znajomością metod programowania matematycznego poważną trudność. Wielkości te, prawie zawsze, określane są heurystycznie. W tym wypadku implementacja w systemie reguł stosowanych przez eksperta dla ich wyznaczenia moze być bardzo efektywna i znacznie ułatwić pracę uzytkownikowi.

3.3 Diagnostyka i ocena rozwiązania

Większość produkowanego obecnie oprogramowania jest wyposażona w odpowiednie moduły (interfejsy) wyprowadzające komunikaty o błędach i zlecenia pozwalające na wyświetlenie dodatkowych informacji (HELP). Często jednak podawany tekst zawiera niewiele więcej niż stwierdzenie błędu lub ogólne sklasyfikowanie sytuacji (np. nadmiar w wyrazeniu

arytmetycznym). W przypadku zadań optymalizacji dodatkową trudność przy informowaniu o wyniku działania programu stwarzają szczególne własności zadań programowania matematycznego. I tak np. w przypadku występowania ekstremów lokalnych można uzyskać rezultat, który mimo spełnienia założonych ograniczeń nie stanowi poszukiwanego rozwiązania. Prawidłowa ocena wymaga w takiej sytuacji podejścia kompleksowego i odwołania się do różnych rodzajów wiedzy, w tym także typu nienumerycznego. Inteligentny program powinien nie tylko podać wynik ostateczny ale uzasadnić konkluzję oraz wskazać w jaki sposób uzyskał rezultat końcowy. Potwierdzenie, że uzyskane rozwiązanie stanowi optimum można uzyskać, między innymi, przez:

- stwierdzenie dostatecznej zmiany wartości pochodnej funkcji celu (w porównaniu z wartością początkową),
- uzyskanie podobnego lub takiego samego rozwiązania dla innych wartości parametrów,
- uzyskanie takiego samego rozwiązania przy przyjęciu innych wartości początkowych,
- zbadanie wpływu niewielkiej zmiany modelu (co powinno spowodować niewielkie zmiany w rozwiązaniu).

Ponieważ obliczenia potrzebne dla potwierdzenia wyników mogą być czasem bardzo pracochłonne, system powinien swoją akcję skonsultować z użytkownikiem.

Przedstawione powyżej przykłady możliwości wykorzystania S.W. na etapie opracowywania modelu, wyboru metody rozwiązania, określania parametrów, a także diagnostyki i oceny rozwiązania nie wyczerpują zagadnienia. Wspólną cechą jest przy tym fakt, że potrzebna wiedza jest znana, a główną trudność stanowi skoordynowanie wysiłków ekspertów z dziedziny optymalizacji, informatyki i inżynierii wiedzy. Ważną rolę w systemie spełnia moduł komunikacji, który umożliwia komunikację człowiek-komputer na poziomie zbliżonym do użytkownika (język naturalny, słowa kluczowe, system menu), nie ulega natomiast zmianie fragment programu realizujący algorytm optymalizacyjny.

Możliwości wykorzystania wiedzy na etapie rozwiązywania zadania optymalizacji wydają się szczególnie interesujące gdyż właśnie tu można uzyskać największy postęp jakościowy. Zagadnienie to wiąże się z pojęciem tzw. systemów sprzężonych (coupled systems), łączących

zalety tradycyjnych systemów obliczeniowych w zakresie analizy numerycznej z osiągnięciami uzyskanymi w dziedzinie systemów wiedzy. Pomimo, ze systemy analizy numerycznej dysponują znacznymi mozliwościami w zakresie przetwarzania danych liczbowych, ich efektywność moze być poprawiona dzięki wykorzystaniu metod inzynierii wiedzy gdyz, jak stwierdzono [34] "mała ilość wiedzy moze czasem oszczędzić ogromną ilość obliczeń". Systemy sprzężone stwarzają takze mozliwość rozwiązywania problemów, które do tej pory z trudnością poddawały się analizie numerycznej jak np. problemy z niepewnymi wartościami danych początkowych. W przypadku zagadnień optymalizacji zastosowanie wiedzy na etapie rozwiązywania zadania programowania matematycznego prowadzi do opracowania algorytmów wykorzystujących wiedzę (knowledge - based algorithms). Wykorzystuje się przy tym fakt, ze w opisie zadania programowania matematycznego (modelu) zawarte są pewne ogólne (globalne) informacje o charakterze jakościowym, które odpowiednio wykorzystane mogą znacznie ułatwić i skrócić etap analizy numerycznej, a w pewnych szczególnych wypadkach pozwalają nawet na znalezienie rozwiązania bez konieczności wykonywania obliczeń.

3.4 Algorytmy wykorzystujące wiedzę

W pracy Papalambrosa [32] wyróżniono trzy rodzaje wiedzy, która moze być wykorzystana w procesie rozwiązywania zadania optymalizacji:
(i) *wiedza globalna* dotyczy faktów niezaleznych od konkretnego położenia punktu (reprezentującego rozwiązanie) i zawiera stwierdzenia słuszne dla wszystkich punktów w przestrzeni projektowej. Większość tego typu wiedzy jest wynikiem doświadczenia i analizy.
(ii) *wiedza lokalna* zawiera stwierdzenia słuszne w odniesieniu do określonego położenia w przestrzeni projektowej. Wiedza ta jest zazwyczaj wynikiem obliczeń.
(iii) *wiedza ewoluująca* obejmuje fakty stwierdzone w wyniku oceny szeregu punktów projektowych (rozwiązań) wygenerowanych przez algorytm. Większość tej wiedzy uzyskiwana jest w wyniku analizy numerycznej.
Typowym przykładem wiedzy globalnej są informacje o monotoniczności funkcji celu czy aktywności ograniczeń. Konkretne wartości funkcji, gradientów czy mnozników są przykładem wiedzy lokalnej. Wiedza ewoluująca powstaje zazwyczaj w wyniku "zakumulowania" informacji

otrzymywanych w wyniku kolejno wykonywanych iteracji (uogólnienie wiedzy lokalnej) jak np. określenie charakteru zmiany wartości funkcji celu czy ograniczeń (zygzakowanie).

Zródłem wiedzy globalnej moze być tzw. analiza monotoniczności (monotonicity analysis). Anlizę monotoniczności jako sposób identyfikacji ograniczeń aktywnych w zadaniach programowania matematycznego zastosował Wilde [43]. Metoda ta była rozwijana przez Wilde i Papalambrosa [27, 28], a następnie przez Papalambrosa i współpracowników [3,4,20-23, 29-33, 42]. Ponieważ analiza monotoniczności (A.M.) jako źródło wiedzy o zadaniach optymalizacji występuje w szeregu prac, ponizej przedstawiono pokrótce podstawowe pojęcia i koncepcje metody.

3.4.1 Analiza monotoniczności

Istotnym celem analizy monotoniczności jest wyznaczenie ograniczeń aktywnych w zadaniu optymalizacji. Pozwala to zazwyczaj, zawęzić obszar poszukiwań i zmniejszyć ilość sprawdzanych warunków dzięki wyeliminowaniu ograniczeń nieaktywnych. A.M. ułatwia także ocenę i interpretację modelu optymalizacyjnego na drodze jakościowej analizy zadania "na poziomie symbolicznym". Zasadnicze znaczenie w praktycznym stosowaniu podejścia mają tzw. reguły monotoniczności [1,33].

Reguła 1 Jezeli funkcja celu jest monotoniczna ze względu na zmienną to istnieje przynajmniej jedno aktywne ograniczenie, które ogranicza zmienną w kierunku przeciwnym do kierunku zmiany funkcji celu. (Ograniczenie jest aktywne względem zmiennej jezeli ogranicza zakres zmienności zmiennej.)

Reguła 2 Jezeli zmienna nie występuje w funkcji celu wówczas musi być albo ograniczona od góry i od dołu przez aktywne ograniczenia lub tez zmienna nie jest ograniczona (ograniczenia dotyczące tej zmiennej są nieaktywne albo nieznaczące.)

Reguły te reprezentują niezbędne warunki optymalności dla układu monotonicznego i odpowiadają szczególnemu przypadkowi warunków KKT dla zadań programowania nieliniowego z ograniczeniami.

Powyzsze podstawowe zasady zostały uzupełnione o trzecią [44],

ułatwiającą analizę w przypadku zadań optymalizacji z nadmiarem ograniczeń.

Reguła 3 Liczba niezredukowanych ograniczeń aktywnych nie może przekroczyć całkowitej liczby zmiennych decyzyjnych.

Podstawowe reguły są uzupełniane dodatkowymi zasadami typu heurystycznego, wynikającymi z analizy monotoniczności funkcji celu i ograniczeń. Przykładem takiej zasady jest tzw. reguła redukcji [1] (Monotonicity Redundancy Principle):

Reguła Redukcji: Jeżeli minimalizowana (maksymalizowana) funkcja celu jest monotoniczna ze względu na zmienną x_i, to jest mało prawdopodobne aby była ograniczona przez nierówność $g_j < 0$, która ma taką samą (przeciwną) monotoniczność ze względu na x_i. Jeżeli ograniczenie ma taką samą (przeciwną) monotoniczność jak minimalizowana (maksymalizowana) funkcja celu ze względu na wszystkie zmienne decyzyjne to można wstępnie założyć, że ograniczenie jest nieaktywne.

W pracy [33] podano przykład reguł stosowanych przy optymalizacji elementu mechanicznego:

Reguła (i) Jeżeli monotoniczność funkcji ze względu na zmienną decyzyjną jest niepewna, ze względu na występowanie członów rosnących i malejących, należy wyeliminować zmienną z jednego z dwóch członów wykorzystując inne ograniczenie aktywne.

Reguła (i+1) Jeżeli ograniczenie nierównościowe określa (dotyczy) słuszność modelu, wówczas dla każdego rozwiązania dopuszczalnego ograniczenie to musi być nieaktywne.

Reguła (i+2) Jeżeli zmienna ma "zwykłe" ograniczenia od góry lub od dołu (tzn. o postaci x<"liczba"), to można założyć że ograniczenia te są nieaktywne.

Wykorzystawanie reguł heurystycznych ma często charakter tzw. wnioskowania defaultowego (prawo brakującej racji). Uproszczenie modelu uzyskane w wyniku zastosowania takiej reguły przyjmuje się za obowiązujące do momentu gdy wygenerowane rozwiązanie nie jest w sprzeczności z zastosowaną heurystyką. I tak w przypadku eliminacji ograniczeń nieistotnych zakłada się, że heurystka jest słuszna do chwili gdy uzyskane wartości nie przekroczą ograniczenia uznanego za nieistotne. Przykład zastosowania A.M. podano w punkcie 4 opracowania,

w którym omówiono przykłady realizacji programów komputerowych.

Istotną cechą A.M. jest wykorzystywanie informacji globalnych, słusznych dla badanego modelu w całym obszarze projektowym, stąd też określenie globalna analiza monotoniczności. Podejście to okazało się efektywne w przypadku stosunkowo małych zadań, a w niektórych szczególnych przypadkach [31], pozwoliło wyznaczyć rozwiązanie bez konieczności wykonywania obliczeń. Dla większych zadań pojawia się jednak szereg trudności związanych z przetwarzaniem symbolicznym, które ograniczają stosowalność metody, zwłaszcza z punktu widzenia automatyzacji obliczeń. Prace nad automatyzacją metody doprowadziły do opracowania tzw. lokalnej analizy monotoniczności [23], wraz z odpowiednimi regułami o charakterze lokalnym tzn. obowiązującymi w otoczeniu analizowanego punktu. W implementacjach komputerowych podstawowe reguły, zarówno globalne jak i lokalne, uzupełniane są szeregiem reguł heurystycznych. Zwiększa to znacznie ogólną liczbę reguł, i tak w programie komputerowym omówionym w pracy [35], wykorzystuje się łącznie 222 reguły. Efektywne wykorzystanie A.M. wymagało więc zastosowania nowych, odmiennych od tradycyjnych, technik reprezentowania i przetwarzania wiedzy. W praktyce, szczególnie wygodne okazały się systemy regułowe; formalizm reprezentowania wiedzy charakterystyczny dla inżynierii wiedzy. Sposób ten zastosowano w ograniczonym zakresie w pracy [4], a rozwinięto w pracach [21, 22, 23]. Przyjęty sposób postępowania polegał na zastosowaniu w pierwszej kolejności wiedzy globalnej, reprezentowanej w postaci szeregu reguł, co pozwalało wyeliminować pewne ograniczenia jako nieaktywne. Następnie wykorzystywano wiedzę lokalną co umożliwiało dalsze ograniczenie zakresu poszukiwań, a dalej wykonywano iteracje. Algorytm postępowania [21, 22] wyróżnia następujące główne etapy:

1. Inicjalizacja i wprowadzenie wiedzy globalnej, k=0.
2. W punkcie x_k zaktualizawanie zbioru roboczego (ograniczeń) wykorzystując reguły zawierające wiedzę globalną i lokalną.
3. Rozwiązanie zadania programowania matematycznego:
   - jeżeli zachodzi przekroczenie ograniczeń należy powrócić do etapu 2 przyjmując k=k+1,
   - jeżeli rozwiązanie zaktualizawanego zbioru spełnia ograniczenia początkowe przejść do następnego etapu.

4. Sprawdzenie kryterium zbieżności i oszacowanie mnożników (związane z wybraną w pracy metodą rozwiązywania zadania optymalizacji). Jeżeli warunki są spełnione obliczenia są zakończone, w przeciwnym wypadku należy powrócić do etapu 2.

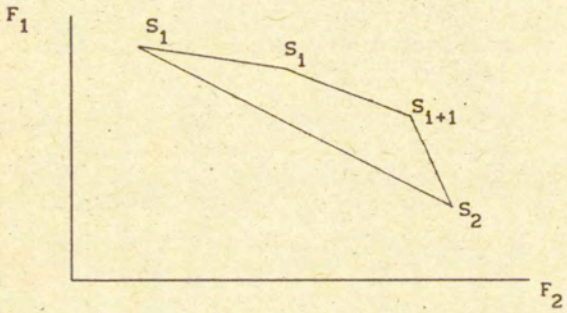
W etapie 2 wykorzystywane są reguły dotyczące globalnej i lokalnej aktywności. Informacje globalne pozwalają sklasyfikować zagadnienie (np. w zależności od tego czy funkcja celu jest monotoniczna ze względu na wszystkie zmienne decyzyjne czy tylko niektóre; sprawdzenie, na podstawie reguł monotoniczności, aktywności ograniczeń i wskazanie związków do ewentualnego wyeliminowania). W omawianym programie wiedza globalna, a także lokalna, reprezentowane były w FORTRANIE. Odpowiednia procedura programu, na podstawie zgromadzonej wiedzy, modyfikuje zadanie, które następnie jest rozwiązywane iteracyjnie. Wadą podejścia jest szerokie wykorzystywanie heurystyk dotyczących analizy "na poziomie lokalnym" co może spowodować że algorytm będzie miał tendencję do wyznaczania rozwiązań "zgodnych" z wprowadzonymi heurystykami pomijając pewne istotne wyniki unikalne.

W pracach [6,12] podano przykład wykorzystania wiedzy dla poprawienia efektywności procedur rozwiązujących zadania optymalizacji wielokryterialnej. Dodatkowa wiedza, reprezentowana jest w postaci odpowiednich reguł (por. rysunek 3.1).

IF $S_1\ S_{i+1}$ *jest najdluzszym niezbadanym segmentem*
AND $S_i S_{i+1}$ *jest wieksze od minimalnej odleglosci dopuszczalnej*
THEN *generuj nastepne rozwiazanie pomiedzy* $S_i\ S_{i+1}$.
IF $S_i\ S_{i+1}$ *jest najdluzszym niezbadanym segmentem*
AND $S_i S_{i+1}$ *jest mniejsze od minimalnej odleglosci dopuszczalnej*
THEN *zakoncz generowanie rozwiazan*,
:
IF *to samo rozwiazanie optymalizuje obie funkcje kryterium*
THEN *nie ma konfliktu pomiedzy kryteriami*.

gdzie minimalna odległość dopuszczalna jest przyjmowana heurystycznie jako ułamek odległości pomiędzy S_1 i S_2.

Opracowany program ma takze mozliwość zmiany metody rozwiązywania zadania optymalizacji w przypadku gdy początkowo wybrana metoda zawodzi, a nowa procedura wywoływana jest automatycznia w trakcie iteracji. Cytowane wyniki numeryczne wskazywały na wyraźną poprawę uzyskiwanych dzięki takiemu podejściu rezultatów.



S_1, S_{i+1} rozwiązania zadania optymalizacji wielokryterialnej,
S_1, S_2 optymalne wartości funkcji F_1, F_2 wyznaczone jak dla zadań jednokryterialnych.

Rys 3.1 Zadanie optymalizacji dwukryterialnej. Funkcje kryteriów F_1, F_2

4. REALIZACJE KOMPUTEROWE

Ponizej omówiono pokrótce programy optymalizacji wykorzystujace wybrane techniki lub metody stosowane w inznierii wiedzy i systemach ekspertowych. Ograniczono się przy tym do systemów, które mogą być uzyte w optymalizacji elementów i układów konstrukcyjnych i mechanicznych pomijając programy zorientowane na inne obszary zastosowań. Przedstawiono takze prototypowy program wspomagający projektowanie konstrukcji, w którym działanie modułów optymalizacji jest kontrolowane przez powłokę ekspertową.

4.1. System EMP

System EMP (Expert System for Mathematical Programming) opracowany został w Instytucie Matematyki w Bayreuth przez K.Schittkowskiego [37]. EMP jest systemem interaktywnym wspomagającym uzytkownika przy tworzeniu modelu, wyborze metody rozwiązania i przygotowaniu danych dla zadań programownia matematycznego z ograniczeniami. Oprócz zadań programowania liniowego, kwadratowego i nieliniowego system moze być stosowany do rozwiązywania problemów optymalizacji wielokryterialnej, minimaksowych, sterowania optymalnego i innych (ogółem 11 obszarów zastosowań), a do dyspozycji uzytkownika znajdują się 24 procedury generujące rozwiązania numeryczne. Funkcja celu i ograniczenia wprowadzane są na drodze interaktywnej, w postaci sekwencji wyrażeń napisanych w języku FORTRAN. W przypadku gdy funkcja celu i ograniczenia są funkcjami liniowymi lub kwadratowymi, wprowadzanie danych sprowadza się do podania odpowiednich elementów macierzy i wektorów. Gradienty funkcji nieliniowych są określane numerycznie, chociaz mogą być takze podane przez uzytkownika w formie symbolicznej. Wygenerowane zadania są zapisywane w bazie danych do późniejszego wykorzystania. EMP dobiera automatycznie właściwy dla danego zadania algorytm rozwiązania i tworzy, na podstawie wprowadzonego opisu, kod programu w języku FORTRAN o formacie zgodnym z wymaganiami biblioteki procedur matematycznych rozwiązujących zadanie iteracyjnie. Uzytkownik moze dołączyć do kodu źródłowego programu własne moduły, a system zachowuje uzyskane wyniki w zbiorze rozwiązań.

System ma mozność uczenia się w ograniczonym zakresie. Dotyczy to głównie efektów zastosowania określonych algorytmów do konkretnych zadań. Tak więc, w miarę wzrostu liczby wykonanych zadań, system

będzie coraz trafniej dobierał algorytmy odpowiednie dla rozwiązania wprowadzonego modelu. W przypadu braku pozytywnego wyniku obliczeń wyprowadzane są komunikaty o przyczynach awarii i sugerowane są dalsze akcje.

Sterowanie pracą systemu realizowane jest za pośrednictwem komend wyświetlanych w formie menu na ekranie monitora.

Zlecenia dotyczą podstawowych czynności jak wprowadzenie (edycja), rozwiązywanie, wyświetlanie wyników ale także przesyłania danych pomiędzy zadaniami, tworzenia zapisu (protokołu) z przebiegu sesji itp. Umozliwia to wyprowdzenie zleceń odpowiadających komendom systemu operacyjnego bez konieczności wychodzenia z EMP. Założeniem autora było stworzenie systemu, który odciąży uzytkownika od czynności wymaganych w tradycyjnych systemach optymalizacyjnych jak np.:

- określenie zbioru dla zadania, inicjacja potrzebnych zbiorów,
- wybór algorytmu i dobór parametrów,
- wczytanie potrzebnych zbiorów,
- deklarowanie dodatkowych obiektów,
- przedstawienie wprowadzanych wyrazeń w postaci wymaganej przez stosowane procedury,
- wstępna interpretacja komunikatów.

W pracy [37] przedstawiono następujące obszary zastosowań, w których system EMP moze być szczególnie przydatny. Obejmują one:

- tworzenie środowiska dla automatycznego fomułowania i rozwiązywania zadań programowania matematycznego,
- badanie modeli optymalizacyjnych (dokonując modyfikacji modelu, a następnie rozwiązując zadanie mozna dobrać wersję najlepiej odpowiadającą badanemu problemowi),
- badanie przydatności algorytmów i ich modyfikacji dla rozwiązania określonych klas problemów,
- gromadzenie wyników numerycznych dotyczących określonych metod (lub zadań).

EMP pozwala uzytkownikowi skoncentrować się na czynnościach związanych z formułowaniem problemu, automatyzując większość dodatkowych czynnościch. W EMP wykorzystano język SUSY, opisany w pracy [39], system aktualnie eksploatowany jest na komputerach VAX, pod systemem operacyjnym VMS lub UNIX.

4.2 System LAGRANGE

System LAGRANGE [38] przeznaczony jest do wspomagania uzytkownika przy rozwiązywaniu zadań optymalizacji z zakresu mechaniki konstrukcji uwzględniając fakt, ze dla modelowania układu wykorzystano metodę elementów skończonych. Jako kryterium optymalizacji przyjmuje się masę układu. Ograniczenia mogą dotyczyć:

- przemieszczeń wezłów,
- odkształceń,
- naprężeń,
- stateczności,
- flutteru,
- częstości drgań własnych,
- przemieszczeń i naprężeń dynamicznych.

Zmienne decyzyjne obejmują:

- przekroje elementów w układach kratowych i ramowych,
- grubości ścian elementów membranowych i płytowych,
- grubości płyt laminowanych w elementach kompozytowych,
- kąty pomiędzy warstwami w elemetach kompozytowych,
- współrzędne węzłów,
- masy balansowe (w optymalizacji z uwzględnieniem flutteru).

Praca z systemem odbywa się w trybie interaktywnym, przy wykorzystaniu systemu menu. Podstawowe opcje umozliwiają, między innymi:

- wprowadzenie opisu nowego zadania,
- zmodyfikowanie istniejących zbiorów,
- zainicjowanie optymalizacji,
- operacje na zbiorach danych,
- sporządzenie dokumentacji przebiegu,
- przeszukanie bazy danych.

System dysponuje zestawem modułów optymalizacyjnych, który pozwala na dobranie algorytmu najbardziej efektywnego dla rozwiązania rozwazanego zadania. W obecnej wersji uzytkownik ma do dyspozycji 8-m procedur realizujących rózne algorytmy, między innymi, metodę SLP (sequential linear programming), metody SQP1, SQP2 (sequential quadratic programming), metodę gradientów. Wyboru procedury dokonuje się wskazując odpowiednią opcję lub tez decyzję pozostawia się systemowi. W tym drugim wypadku, odpowiedni moduł w wyniku interakcji z

uzytkownikiem i wykorzystując odpowiednie reguły, zaproponuje właściwy algorytm. Uzytkownik moze zaakceptować decyzję lub tez dokona własnego wyboru. Dla duzych zadań optymalizacja w trybie konwersacyjnym jest mało wygodna dlatego tez istnieje mozliwość przejścia na pracę wsadową.

LAGRANGE wyposazony jest w elastyczne moduły ułatwiające interpretację i ocenę sytuacji awaryjnych. Jednocześnie wyświetlany jest tekst komunikatu sugerującego właściwą akcję.

System stosowany był praktycznie dla optymalizacji duzych układów konstrukcyjnych, między innymi, przy projektowaniu pionowego stabilizatora Airbusa A310.

4.3 System OPTIMA

OPTIMA przedstawiany jest jako system optymalizacyjny ogólnego przeznaczenia zorientowany na wykorzystanie w zagadnieniach inzynierskich [5]. Opracowany został na Wydziale Architektury Universytetu w Sydnej. Przy opracowywaniu systemu sformułowano następujące wymagania:

- łatwy i szybki sposób opisywania problemu,
- reprezentowanie i przekazywanie informacji o zadaniu w naturalny sposób,
- mozliwości modyfikowania zadania,
- umiejętność sformułowania, na podstawie opisu, zaleznośći funkcyjnych określających funkcję celu i ograniczenia,
- zdolność rozpoznawania typów wyrazeń arytmetycznych,
- umiejętność doboru właściwej metody rozwiązywania zadania optymalizacji i automatycznego wykonania obliczeń.

Na podstawie pseudo-naturalnego opisu, system tworzy układ związków matematycznych opisujących rozwiązywane zagadnienie. Następnie odpowiedni moduł, identyfikuje rodzaj zadania i dobiera odpowiednią metodę rozwiązania oraz inicjuje obliczenia. System szeroko wykorzystuje reprezentację regułową i ramkową. Przykład reprezentacji ramkowej, w przypadku optymalizacji rozkładu pomieszczeń w mieszkaniu, przedstawiono na rysunku 4.1.

```
nazwa:                          kuchnia
slots:      rodzaj
                wartosc:        prostokąt
            dlugosc
                wyrazenie:      (dlugosc_kuchni)
            szerokosc
                wartosc:        8.0
                jednostki:      m
            koszt_jednostowy
                wartosc:        200
```

a) Opis kuchni

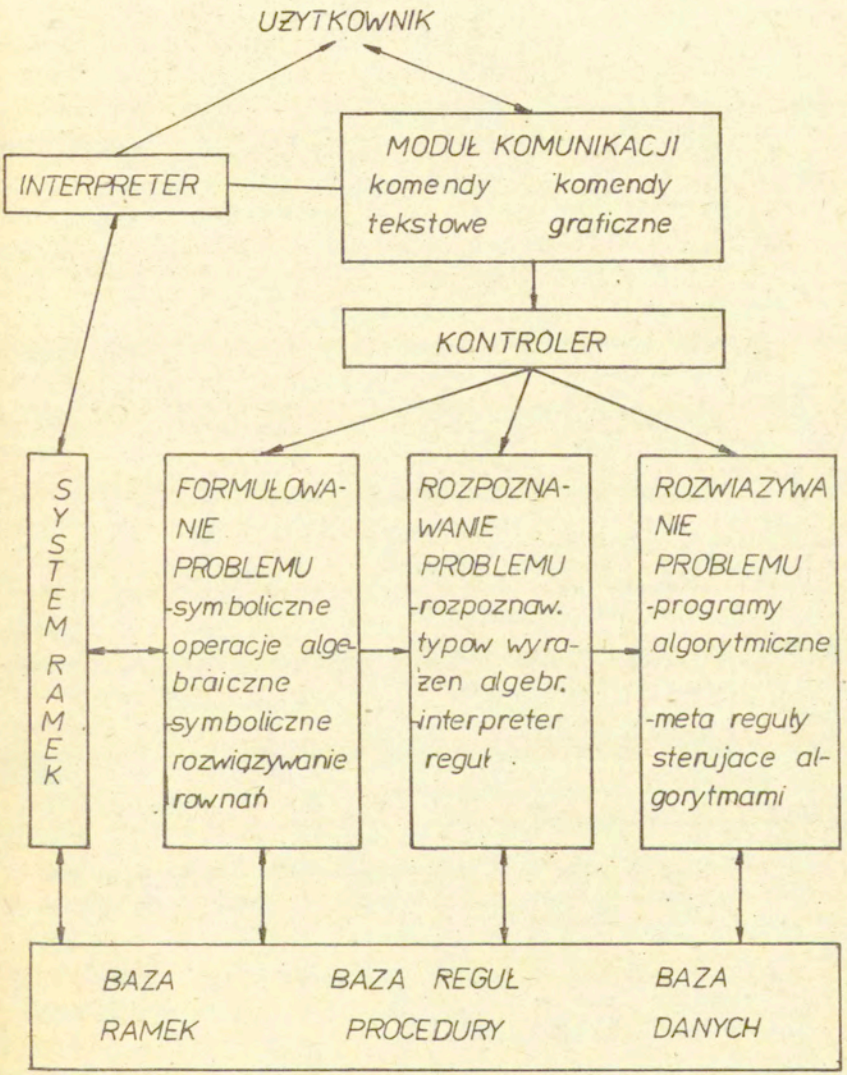
```
nazwa:                          kryterium_1
slots:      funkcja_celu
                wyrazenie:      (powierzchnia_domu)
            kryterium_optymalizacji
                wartosc:        maksymalizacja
```

b) Reprezentacja kryterium optymalizacji

```
nazwa:                          ograniczenie_1
slots:      lewostronne
                wyrazenia:      (pole_przekroju)
            prawostronne
                wartosc:        20.0
                jednostki:      m*m
            typ
                nierownosciowe: mniejsze_lub_rowne
```

c) Reprezentacja ograniczenia

Rysunek 4.1 System OPTIMA. Reprezentacja ramkowa



Rys 4.2 System OPTIMA. Schemat programu

Sformułowanie odpowiednich zależności matematycznych opisujących rozwiązywane zagadnienie związane jest, przede wszystkim z symbolicznymi operacjami algebraicznymi. W systemie OPTIMA ten fragment pracy wykonuje odpowiedni moduł napisany w języku LISP, korzystający z odpowiedniej wiedzy zapisanej w formie reguł.

Aby dobrać właściwą metodę rozwiązania zadania optymalizacji system musi, w pierwszej kolejności, rozpoznać zmienne i zależności pomiędzy nimi. Ten fragment analizy wykonuje moduł napisany w PROLOGu.

Część obliczeniową (optymalizację matematyczną) realizują moduły napisane w języku C.

Ogólny schemat systemu przedstawiono na rysunku 4.2 [5]. Podstawowym elementem systemu jest kontroler (communication controller), który steruje przepływem informacji pomiędzy trzema modułami składowymi:

- modułem formułującym zadanie,
- modułem rozpoznającym,
- modułem rozwiązującym.

System OPTIMA zaimplemetowany jest na stanowisku typu SUN.

4.4 System SYMON-SYMFUNE

System SYMON - SYMFUNE [1] opracowany został na Wydziale Mechanicznym UCLA. Program prowadzi analizę na trzech poziomach abstrakcji, odpowiadających ludzkiemu sposobowi rozumowania.

Poziom I - wnioskowanie jakościowe.

Odpowiada wnioskowaniu o obiektach i ich własnościach w taki sposób, że nie jest to związane z ilościowym opisem problemu. Przykładem wnioskowania jakościowego jest np. stwierdzenie, że energia potencjalna sprężyny wzrasta wraz z odkształceniem od położenia równowagi czy obserwacja, że naprężenie styczne w skręcanym elemencie o przekroju walcowym, maleje wraz ze wzrostem grubości ścianki. Istotne jest przy tym odkrycie tendencji zmian, a nie określenie konkretnej zależności. (Ten poziom analizy, dotychczas zastrzeżony dla człowieka, jest obecnie dziedziną, w której próbuje się zastosować techniki sztucznej inteligencji. W przypadku realizacji komputerowej odpowiednie moduły dokonujące analizy jakościowej są zazwyczaj pisane w językach wysokiego poziomu opartych często na typowych językach Sz. I. jak np. LISP.)

Poziom II - wnioskowanie funkcyjne.

Prace nad automatyzacją wnioskowania funkcyjnego zmierzają do stworzenia programów, które w wyniku analizy symbolicznej byłyby w stanie podawać rozwiązania w postaci funkcyjnej. Podstawowym aparatem są tu algorytmy i techniki do przetwarzania symbolicznego. Przykładem wnioskowania funkcyjnego jest np. stwierdzenie, ze energia potencjalna sprężyny zmienia się proporcjonalnie do kwadratu odległości od położenia równowagi.

Poziom III - wnioskowanie ilościowe (analiza numeryczna).

Na tym poziomie wyznacza się zależności ilościowe pomiędzy obiektami. Jest to etap, na którym komputery są juz powszechnie wykorzystywane.

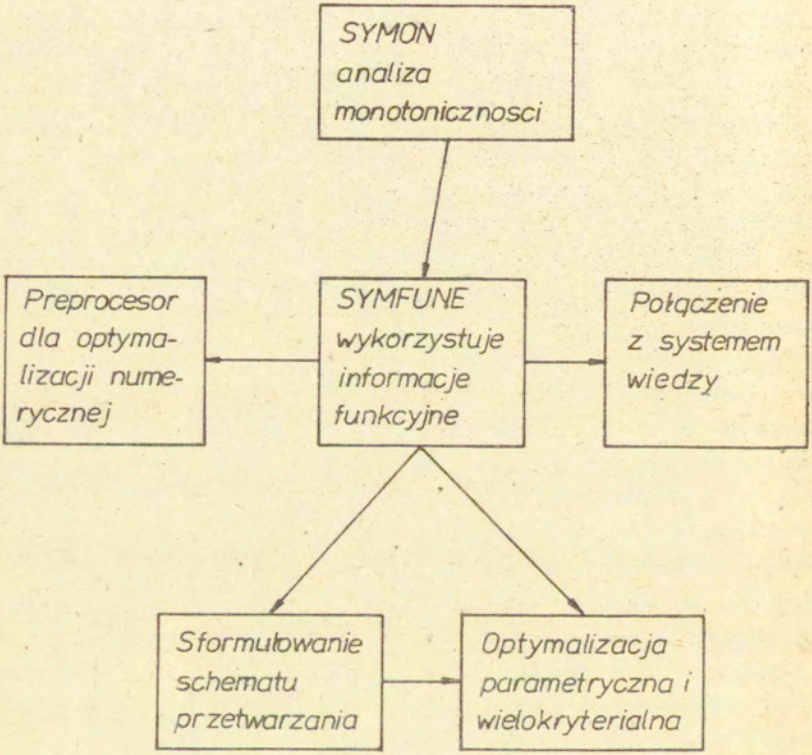
Oddzielne moduły systemu SYMON-SYMFUNE prowadzą analizę na trzech wymienionych poziomach:

- moduł SYMON (SYmbolic MONotonicity analysis) dokonuje analizy monotoniczności (analiza jakościowa),
- moduł SYMFUNE (SYMbolic FUNctional Evaluation) wykonuje optymalizację symboliczną,
- moduł FORTRANIZER przekształca związki wygenerowane przez moduł SYMFUNE na kod programowy w języku FORTRAN, który następnie moze być rozwiązany numerycznie.

Na rysunku 4.3 przedstawiono schematycznie przekazywanie sterowania w systemie SYMON-SYMFUNE.

Moduł SYMON

Stanowi samodzielny program napisany w języku VAXIMA przeznaczonym do obsługi matematycznych operacji symbolicznych. Jest to język wysokiego rzędu napisany w FranzLISPie [1], stosowany na komputerach VAX pod systemem operacyjnym UNIX. Dane wejsciowe do programu stanowią: funkcja celu (minimalizowana) oraz ograniczenia liniowe i nieliniowe. Program SYMON dokonuje symbolicznego różniczkowania aby wyznaczyć poszukiwane monotoniczności (+ lub -). Dla wygody uzytkownika wyniki różniczkowania wyprowadzane są w postaci tzw. tablicy monotoniczności. W kolumnach tablicy umieszcza się znaki + i - odpowiadające monotonicznościom funkcji celu i ograniczeń ze względu na zmienne decyzyjne. Następnie stosując regułę redukcji oraz reguły monotoniczności (opisane w p. 3.4.1) wyznacza się ograniczenia aktywne.



Rys 4 3 System SYMON-SYMFUNE

Proces ten odbywa się następująco: na wstępie zakłada się, ze kolejne ograniczenia są nieaktywne. Jezeli dla tak zmodyfikowanego układu nie można wyznaczyć rozwiązania oznacza to, ze analizowane ograniczenie musi być aktywne. W wyniku wykonania tego etapu, otrzymuje się alternatywne zbiory ograniczeń aktywnych. Schemat realizacji analizy monotonicznosci w module SYMON przedstawiono na rysunku 4.4. Ostatecznym wynikiem działania programu jest lista zawierająca alternatywne kombinacje aktywnych i nieaktywnych ograniczeń. Kombinacje nie wyszczególnione w wyniku analizy monotoniczności nie będą uwzględniane w dalszych etapach. W szczególnych wypadkach (for constrained bounded problems) SYMON może podać rozwiązanie w postaci funkcyjnej.

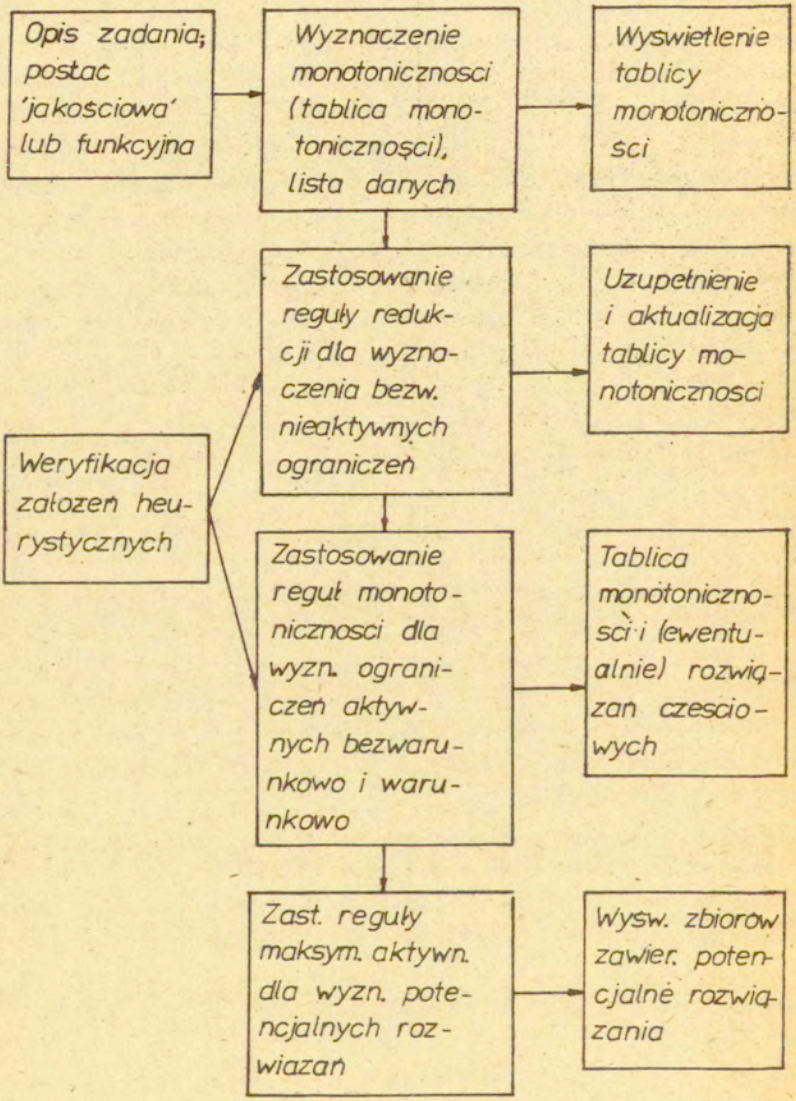
Moduł **SYMFUNE**

Obszar, w którym poszukuje się rozwiązań został zawężony dzięki analizie dokonanej w programie SYMON i można by już przystąpić do obliczeń numerycznych, system prowadzi jednak dalej analizę na poziomie symbolicznym. W zależności od rodzaju badanego problemu stosowane są różne strategie postępowania;

- Jezeli rozwiązanie jest "ograniczone" (constrained -bounded) można oczekiwać, ze uda się wyznaczyć rozwiązanie w postaci parametrycznej,
- Jezeli rozwiązanie nie jest "ograniczone" funkcja celu jest przekształcana przy wykorzystaniu aktywnych ograniczeń. Następnie przyrównuje się do zera pochodne funkcji celu, a uzyskane równania pozwalają na wyznaczenie (lub wyeliminowanie) pozostałych zmiennych decyzyjnych. Gdy zmodyfikowana funkcja celu jest niezalezna od pozostałych zmiennych oznacza to, ze istnieje nieskończenie wiele rozwiązań, a zmienne decyzyjne muszą spełniać ograniczenia uznane za nieaktywne. (Należy je ponownie włączyć do rozważanego zestawu związków.)

Po wyznaczeniu zbiorów zawierających potencjalne rozwiązanie, program dokonuje weryfikacji wykorzystując (zmodyfikowaną) metodę mnożników Lagrangea łącznie z warunkami optymalności KKT. (Np. bezwarunkowo negatywna wartość mnożników Lagrangea wskazuje, ze przypadek nigdy nie będzie optymalny, wartość bezwarunkowo dodatnia wskazuje, ze nie trzeba uzupełniać ograniczeń o dodatkowe warunki.)

Wynikiem działania programu SYMFUNE są funkcje stanowiące parametryczne rozwiązanie zadania optymalizacji lub też związki (nierówności)



Rys 4.4 Modul SYMON, analiza monotonicznosci

stanowiące zmodyfikowane zadanie optymalizacji.

Moduł FORTRANIZER

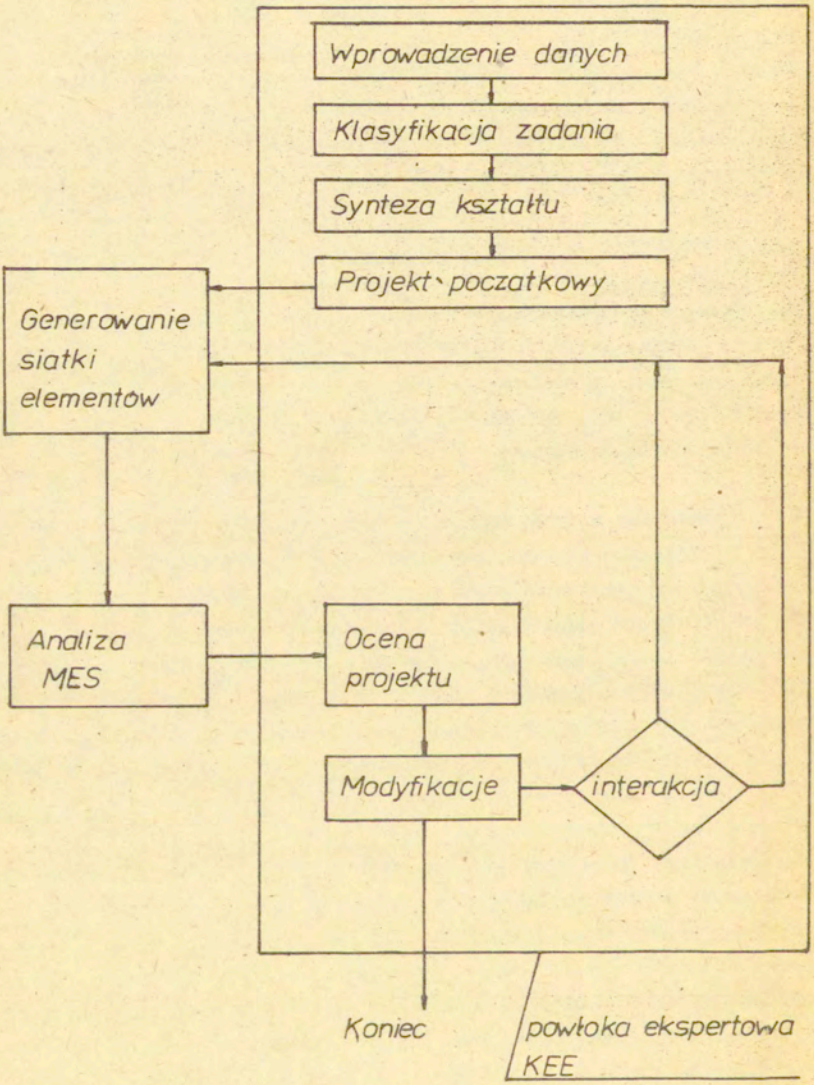
Program ten, napisany w języku VAXIMA tworzy, na podstawie danych wygenerowanych przez SYMFUNE, program w języku FORTRAN, który stanowi opis zadania optymalizacji, w formacie wymaganym przez standardowe procedury programowania matematycznego dokonujące analizy numerycznej.

Zamieszczone w literaturze [1, 2, 3] informacje wskazują, że w niektórych przypadkach zadań inżynierskich z zakresu mechaniki, zastosowanie przedstawionego podejścia umożliwiło uzyskanie rozwiązania zadania optymalizacji w postaci zamkniętej. Jednocześnie wątpliwości budzi możliwość wykorzystania systemu do optymalnego projektowania konstrukcji np. układów wielokrotnie statycznie niewyznaczalnych i o wielu stopniach swobody.

4.5 Prototypowy program wspomagający projektowanie konstrukcji

Program opracowywany jest na Wydziale Mechaniki i Inżynierii Lotniczej Uniwersytetu Rutgersa w New Jersey [14]. Przeznaczony jest do projektowania konstrukcji modelowanych elementami skończonymi tarczowymi (konstrukcje znajdujące się w PSO, PSN, OS). Aktualnie jest to jeszcze prototyp systemu, wydaje się jednak, że wskazuje on jeden z kierunków, w którym może nastąpić rozwój programów inżynierskich.

Istotnym elementem programu jest komercyjna powłoka ekspertowa KEE, która z jednej strony pełni funkcje kontrolne, sterując ogólnym przebiegiem obliczeń, z drugiej zaś, usprawnia i ułatwia realizację obliczeń na poszczególnych etapach dzięki możliwości wykorzystania dodatkowej wiedzy zapisanej w bazie wiedzy. Wiedza wykorzystywana jest do właściwego sklasyfikowania zadania ze względu na typ i kształt konstrukcji, rodzaj obciążenia, własności materiału, zastosowane elementy skończone itp. Również na etapie modyfikowania kształtu konstrukcji wykorzystywane są odpowiednie reguły zakodowane w bazie wiedzy. Modyfikacja obejmuje usunięcie z modelu konstrukcji elementów najmniej wytężonych. Wielkość naprężeń stanowi jedynie podstawę do umieszczenia elementu na liście "kandydatów do usunięcia". Lista ta jest weryfikowana pod względem logicznej poprawności. Sprawdzane jest np. czy usunięcie elementu nie spowoduje zmiany schematu podparcia,



Rys 4.5 Schemat programu

powstania pustej przestrzeni wewnątrz konstrukcji, przerwania spójności itp. Schemat programu przedstawiono na rysunku 4.5.

5. UWAGI KONCOWE

Liczba realizacji programów optymalizacji wykorzystujących metody i techniki inżynierii wiedzy nie jest duża. Nacisk kładzie się przy tym głównie na wyposażenie programu w moduły zapewniające efektywną komunikację człowiek-komputer, w miarę możliwości, mające zdolność przetwarzania języka naturalnego (pseudonaturalnego). Stosuje się także pakiety do operacji symbolicznych ułatwiające wprowadzanie i operowanie wzorami matematycznymi. Jest to widoczna tendencja, znajdująca także potwierdzenie w wypowiedziach w ramach dyskusji (np. [26]). Ponieważ naturalnym trybem pracy takich systemów jest interakcja, moduł komunikacji odpowiedzialny jest za narzucenie odpowiedniej strategii dialogu. Jedną z funkcji modułu jest wyprowadzanie odpowiednich komentarzy i komunikatów, co jest szczególnie istotne w przypadku błędów lub zakończenia realizacji bez uzyskania rozwiązania. Niektóre ze zrealizowanych programów podają w takiej sytuacji sugestię, co należy uczynić aby poprawić wykryty błąd lub niedociągnięcie. Wydaje się, że w niedługim czasie moduły komunikacji posiadające pewne możliwości rozpoznawania języka naturalnego lub przynajmniej pojedynczych wyrazów (komend) oraz wyposażone w mechanizm dobierający teksty komunikatów będą stanowiły standardowe wyposażenie programów użytkowych.

Istotną trudność przy opracowywaniu systemów wiedzy stanowi skoordynowanie wysiłków specjalistów z zakresu inżynierii wiedzy z ekspertami z danej dziedziny. Brak takiej współpracy był widoczny w wielu z prezentowanych w literaturze programach, które w znacznym stopniu były tworzone przez bardzo nieliczne zespoły (lub pojedyncze osoby). Systemy charakteryzujące się dobrymi możliwościami z zakresu optymalizacji stosunkowo słabo stosowały techniki inżynierii wiedzy. Jednocześnie nawet sprawny mechanizm wnioskowania musi dysponować odpowiednio zasobną bazą reguł, reprezentującą właściwy poziom merytoryczny. W takiej sytuacji stworzenie wartościowego

systemu wiedzy jest zadaniem przekraczającym mozliwości jednej osoby.

Przedmiotem szczególnej uwagi były programy stosowane w zagadnieniach optymalizacji konstrukcji. Cechą takich programów, jak zresztą większości produktów stosowanych w inzynierii, jest konieczność wykonywania obliczeń. Program musi być zdolny do przejścia od przetwarzania wiedzy na poziom obliczeń, zachodzi więc potrzeba sprzężenia w jednym systemie obu sposobów przetwarzania. W szczególności wiedza moze być potrzebna jedynie do poprawienia efektywności algorytmu numerycznego. Tworzenie takich programów wiąże się z dodatkowymi trudnościami jak np. wybór języka programowania. Kompilatory stosowane w Sz.I. do reprezentowania wiedzy jak LISP czy PROLOG są mało wygodne przy kodowaniu algorytmów numerycznych, a łączenie modułów napisanych w różnych językach jest często bardzo trudne lub wręcz niemozliwe (np. TURBO PROLOG i TURBO PASCAL).

PIŚMIENNICTWO

[1]. A.M. Agogino, A.S. Almgren: Techniques for integrating qualitative reasoning and symbolic computation in engineering optimization, Eng. Opt., 1987, Vol.12, No 2, str. 117 - 135.

[2]. A.M. Agogino, A.S. Almgren: Symbolic computation in computer-aided optimal design, w: Expert systems in CAD, red: J.Gero, Elsevier Publ., IFIP, 1987, str. 267 -284.

[3]. S.Azarm, P.Papalambros: A case study for knowledge based active set strategy, Trans. ASME, JMTAD, Vol. 106, nr 1, 1984.

[4]. S.Azarm, P.Papalambros: An automated procedure for local monotonicity analysis, Trans. ASME, JMTAD, Vol. 106, nr 1, 1984.

[5]. M.Balachandran, J.S.Gero: A knowledge - based approach to mathematical design modeling and optimization, Eng. Opt., 1987, Vol. 12, No 2, str. 91 - 115.

6]. M.Balachandran, J.S.Gero: Use of knowledge in selection and control of optimization algorithms, Eng. Opt., 1987,Vol 12, No. 2, str. 163 - 173.

[7]. A. Borkowski: Users's Guide for System MATRIX, IPPT PAN, Warszawa 1987.

[8]. J.Cagan, A.M.Agogino: Reasoning about mechanical structures from first principle, Proc. 12th IMACS Congr. on Scientific Computations, Paryż, lipiec 1988.

[9]. R.J.Douglas: A qualitative assessment of parallelism in expert systems, Comm. of ACM, 1985, Vol 28, No. 1,str. 70-81.

[10]. R.Fikes, T.Kehler: The role of frame-based representation in reasoning, Comm. of the ACM, 1985, Vol. 28, No. 9, str 902-941.

[11]. P. Friedland: Special section on architectures for knowledge-based systems, Comm. of the ACM, 1985, Vol. 28, No. 9, str 902 - 941.

[12]. J.Gero, M.Balachandran: Knowledge and design decision process, w: Applications of AI in Engn. Problems, Proc. 1st Int. Conf., red.: D.Sriram, R.Adey, Springer, 1986.

[13]. F.Giannesini, H.Kanoni, R.Pasero, M. van Caneghen: PROLOG, Addison-Weseley, 1986.

[14]. P.K.Hart, J.Rodrigues: A dual-purpose KBES for preliminary structural deisgn, w przygotowaniu.

[15]. D.Hartman: Application of AI tools for re-analysis within structural optimization, Techn. Report, University of Dortmund, Dortmund 1985.

[16]. D.Hartman: Selection and evaluation of structural optimization strategies by means of expert systems, Techn Report, University of Dortmund, Dortmund 1985.

[17]. F.Hayes-Roth, D.A.Watterman, D.B.Lenat: Building expert systems, Addison-Wesley,1983.

[18]. F.Hayes-Roth: Rule-based systems, Comm. of the ACM, 1985, Vol. 28, No. 9, str 902 - 941.

[19]. A.C.Hearn red: REDUCE user's manual The Rand Corporation, Santa Monica 1983.

[20]. H.L. Li, P. Papalambros: REDUCE applications in design optimization, Proc. CAD/CAM, Robotics and Automation, Tucson, luty 1985, str. 187-192.

[21]. H.L.Li, P.Papalambros: A combined local-global active set strategy for nonlinear design optimization, Proc. ASME Design Automation Conference, wrzesień 1987, Boston.

[22]. H.L.Li, P.Papalambros: A production system for use of global optimization knowledge, Trans. ASME, JMTAD, referat nr 84-det 194.

[23]. H.L.Li, P.Papalambros: An interior inear programming algorithm using local and global knowledge, Trans. ASME, JMTAD, referat nr 87-det 4,1987.

[24]. F.A.Lootsma: Comperative performance evaluation, experimental desig and generation of test problems in nonlinear optimization, w: Computational Mathematical Programming, red: K.Schittkowski, NATO ASI Series, Serie F, Vol. 15, 1985, str. 249-260.

[25]. MACSYMA Reference manual, Tech. Report, MIT,Massachusets, 1977.

[26]. A.J. Morris: Potentials of AI methods in optimization, dyskusja panelowa, w: Computer Aided Optimal Design, red: A. Mota Soares, NATO ASI Series, Serie F, Vol. 27, Springer, 1987.
[27]. P.Papalambros D.J.Wilde: Global non-iterative design optimization using monotonicity analysis, J. Mech. Des., ASME, 101(3), 1979, 645-649.str
[28]. P.Papalambros, D.J.Wilde: Regional monotonicity in optimum design, Trans. ASME, J.M.Des., Vol. 102, Nr 3, 1980.
[29]. P. Papalambros: Monotonicity in goal and geometric programming, Trans. ASME, J.M.Des., Vol. 104, 1982.
[30]. P.Papalambros: Qualitative monotonicity analysis of a problem involving defferential equations, Eng. Opt., Vol. 6, 1983, str. 117-128.
[31]. P.Papalambros: Knowledge-based systems in optimal design, w: Computer Aided Optimal Design, red: A.Mota-Soares, NATO ASI Series, Serie F, Vol. 27, Springer 1987, str 311-362.
[32]. P. Papalambros: Integration of knowledge forms in design optimization, Proc. of NSF Workshop on the Design Process, Oakland 1987, red: M.B. Waldron.
[33]. P.Papalambros: Codification of semi-heuristic global processing of optimal design models, Proc. of NSF Workshop on the Design Process, Oakland 1987, red: M.B Waldron.
[34]. J.Pearl: Knowledge versus search: a quantitative analysis using AI A.I., Vol. 20, Nr 1, 1983.
[35]. J.R.Rao, P.Papalambros: Implementation of semi-heuristic reasoning for boundedness analysis of design optimization models, Proc. ASME, Design Automation Conf., wrzesień 1987, Boston.
[36]. A.Ruszczyński, T.Kręglewski, T.Rogowski: MSPN, Prace Instytutu Automatyki PW.
[37]. K. Schittkowski: EMP an expert system for mathematical programming, University of Bayreuth, Bayreuth 1985.
[38]. K.Schittkowski: An integrated knowledge-based problem solving system for structural optimization, w: Structural optimization, red: G.Rozwany, B.L.Karihaloo, Kluver Acad. Press, 1988, str. 289-297.
[39]. K.Schittkowski: Die Systementwicklungssprache SUSY, Report, Mathematisches Institute, Universitaet Bayreuth, Bayreuth 1987.
[40]. C.Townsend: Introduction to TURBO PROLOG, 1987, SYBEX, San Fransisco.
[41]. C.Townsend: Advanced techniques in TURBO PROLOG, 1987, SYBEX, San Fransisco.
[42]. N. Tzannetakis, P. Papalambros: An active set sequential linearization algorithm for nonlinear design optimization, Proc. ASME Design Automation Conf., wrzesień 1987, Boston.
[43]. D.J.Wilde: Monotonicity and dominance in optimal hydraulic cylinder design, Trans. ASME, Vol. 97, 1975, str. 1310-1314.
[44]. D.J. Wilde: A maximal activity principle for eliminating overconstrained optimization cases, Trans. ASME, JMTAD, Vol. 108, 1986, str. 312-314.
[45]. P.H.Winston: Artificial Intelligence, Addison-Wesley, 1984.
[46]. Proc. 2nd ASCE Conf.. on Electronic Computations, Pittsburg, 1960.