Studium Doktoranckie Informatyki

Wydział Automatyki, Elektroniki i Informatyki

Politechniki Śląskiej w Gliwicach

WYBRANE PROBLEMY AUKCJI I PRZETARGÓW KOMBINATORYCZNYCH

mgr inż. Tomasz Tatoń

Promotor:

prof. zw. dr hab. inż. Antoni Niederliński

Gliwice, 2008

Oświadczam, że niniejsza praca jest moim samodzielnym opracowaniem, nie naruszającym niczyich praw autorskich.

Gliwice, 17.10.2008

Tomasz Tatoń

Podziękowanie

Chciałbym w tym miejscu podziękować wszystkim tym, których zainteresowanie i pomoc przyczyniły się do powstania tej pracy. W szczególności mojemu promotorowi Panu Profesorowi Antoniemu Niederlińskiemu, za udzielaną pomoc w trakcie pisania pracy, cenne uwagi i poświęcony czas, mojej żonie Basi za udzielone wsparcie i cierpliwość, a także całej mojej rodzinie za okazaną pomoc podczas trwania studiów doktoranckich.

Spis treści

Spis rysunków

Spis tabel

Rozdział 1

Wstęp

1.1 Zarys aukcji i przetargów

Aukcja jest zorganizowaną formą sprzedaży ofertowej, o z góry ustalonych regułach, prowadzoną w czasie rzeczywistym. Reguły te powinny być znane uczestnikom aukcji (ang. bidders) przed jej rozpoczęciem. Powinny określać między innymi sposób wyznaczenia oferty lub zbioru ofert wygrywających aukcję, a także czas jej zakończenia. Aukcja jest również formą przetargu, prowadzonego w czasie rzeczywistym. Przedmiotem aukcji mogą być takie elementy jak towary, nieruchomości, usługi lub zamówienia na wykonanie określonych prac, oferowane przez agenta lub agentów prowadzących aukcję. Uczestnicy aukcji mogą składać oferty cenowe na poszczególne elementy lub na dowolnie konfigurowane zbiory tych elementów, o ile reguły aukcji na to pozwalają. Oferta cenowa dla takiego zbioru może różnić się od sumy ofert cenowych poszczególnych jego elementów. Aukcja jest kwintesencją gospodarki rynkowej, a mianowicie wartość każdego towaru jest określana przez cenę, którą ktoś jest gotów za ten towar zapłacić.

1.2 Problem wyznaczenia zwycięzcy

Istnieje ścisłe powiązanie aukcji i przetargów z zagadnieniami związanymi z kombinatorycznymi problemami decyzyjnymi. Stąd też wywodzi się określenie *aukcji kombinatorycznej* (ang. Combinatorial Auctions - CAs). Nazwa ta wynika z faktu, że w ogólnym przypadku problem wyznaczenia zwycięzcy aukcji kombinatorycznej (ang. The Winner Determination Problem - WDP) [CSe06] jest decyzyjnym, binarnym problemem kombinatorycznym. Ogólnie

rzecz biorąc problem wyznaczenia zwycięzcy polega na wyborze oferty lub ofert, które maksymalizują dochód ze sprzedaży towarów lub usług, zawartych w tych ofertach.

1.3 Programowanie w logice z ograniczeniami

Programowanie w logice z ograniczeniami (ang. Constraint Logic Programming - CLP) [Nie99] jest metodą modelowania i rozwiązywania zagadnień z ograniczeniami. Oparte jest o podstawowe zależności logiczne modelu i jego ograniczeń. Rozwiązanie problemu uzyskuje się na drodze odpowiedniego opisu problemu i badaniu jego ograniczeń za pomocą elementarnych zależności logicznych. CLP jest również metodą programowania deklaratywnego, nie wymagającą sformułowania algorytmu rozwiązania problemu, a jedynie takiego jego opisu, który jest akceptowalny przez kompilator stosowanego języka CLP. Taki właśnie opis problemu jest często całym programem rozwiązania danego problemu, lub jego zasadniczą częścią.
W ostatnich latach CLP staje się coraz bardziej popularną metodą rozwiązywania różnego typu problemów decyzyjnych. Można do nich zaliczyć między innymi problemy optymalizacji kombinatorycznej, takie jak alokacja środków [Tat06], rozkrój, zagadniania transportowe [Szk07]. Przy pomocy programowania w logice z ograniczeniami można również rozwiązywać problemy spełnienia ograniczeń (ang. Constraint Satisfaction Problem (CSP)) [Bar98]. Do tej grupy problemów można zaliczyć między innymi harmonogramowanie statyczne, polegające na przyporządkowaniu zasobów czynnościom (por. [Leg06]) i harmonogramowanie dynamiczne, polegające na przyporządkowaniu zasobów i czasu czynnościom (por. [SWB05] i [Szc02]).

1.4 Cel i tezy pracy

Głównym celem niniejszej pracy, jest opracowanie efektywnych modeli, metod poszukiwania oraz metod optymalizacji dla ogólnego problemu wyznaczenia zwycięzcy aukcji kombinatorycznej, przy zastosowaniu paradygmatu programowania w logice z ograniczeniami.

Uzupełnieniem celu głównego są następujące cele cząstkowe:

- Zbadanie efektywności czasowej tych modeli.
- Zbadanie wpływu zmienności danych na efektywność czasową modeli.
- Analiza wpływu dodatkowych ograniczeń w modelach CLP na czas poszukiwania rozwiązań problemu WDP.
- Wpływ zmian strategii numeracyjnych na wielkość przestrzeni decyzyjnej oraz na czas poszukiwania rozwiązań problemu WDP.
- Wpływ zmiany metody ukonkretniania zmiennych na czas wyznaczania rozwiązania optymalnego.

W pracy zostały przedstawione i uzasadnione następujące tezy, nie mające odpowiedników w dotychczasowej literaturze przedmiotu:

- Za pomocą programowania w logice z ograniczeniami można zbudować modele umożliwiające efektywne rozwiązywanie problemu wyznaczenia zwycięzcy aukcji kombinatorycznej.
- Istnieje wiele różnych metod modelowania ograniczeń dla problemu wyznaczenia zwycięzcy za pomocą programowania w logice z ograniczeniami.
- Implementacja dodatkowych ograniczeń substytucyjnych i/lub komplementarnych w model CLP reprezentujący problem wyznaczenia zwycięzcy ma znaczący wpływ na czas poszukiwania rozwiązań dopuszczalnych, w tym również rozwiązania optymalnego.
- Zastosowanie programowania w logice z ograniczeniami do rozwiązywania problemu wyznaczenia zwycięzcy aukcji kombinatorycznej umożliwia wykorzystanie dodatkowych metod zmniejszania drzewa decyzyjnego, co ma znaczący wpływ na czas generowania rozwiązań problemu.
- Czas wyznaczenia rozwiązań dopuszczalnych i/lub rozwiązań optymalnych dla modeli CLP nie jest monotoniczną funkcją liczby towarów wystawianych na aukcji.

- Pomiędzy sformułowaniem problemu za pomocą programowania w logice z ograniczeniami, a programem rozwiązującym problem wyznaczania zwycięzcy aukcji kombinatorycznej nie ma *przepaści semantycznej*:
  - program jest formalnym zapisem słownego sformułowania problemu,
  - nie ma potrzeby sprowadzania problemu do jakieś postaci kanonicznej.

1.5 Układ pracy

Pracę rozpoczyna rozdział pierwszy. Zawiera on wstępne informacje dotyczące zagadnień związanych z aukcjami i przetargami, w tym także połączenie aukcji i przetargów z kombinatorycznymi problemami decyzyjnymi oraz wstępny opis zagadnienia związanego z techniką programowania w logice z ograniczeniami. Sformułowano w nim genezę problemu, cel i tezy pracy.

Rozdział drugi przedstawia szczegółowe informacje dotyczące aukcji i przetargów. Zawarty w nim został podział i klasyfikacja aukcji, a także powiązanie ich z kombinatoryką poprzez sformułowanie problemu wyznaczenia zwycięzcy. Rozdział ten przedstawia również modele matematyczne głównych typów aukcji kombinatorycznych, koncentrując się na ich związku z problemem wyznaczenia zwycięzcy. W rozdziale tym znajduje się również szczegółowy opis dla dwóch przykładów, dwóch różnych typów aukcji kombinatorycznych oraz przykłady praktycznych zastosowań aukcji kombinatorycznych.

W rozdziale trzecim przedstawiono obecnie znany stan zagadnienia aukcji kombinatorycznych oraz opisano znane metody rozwiązywania problemu wyznaczenia zwycięzcy przy stosowaniu różnych technik programistycznych.

W rozdziale czwartym przedstawiono szczegółowo zagadnienia związane z paradygmatami programowania w logice i programowania w logice z ograniczeniami, a także przedstawione zostały różnice zawarte pomiędzy tymi paradygmatami. W szczególności omówiony został problem spełnienia ograniczeń z wykorzystaniem propagacji ograniczeń oraz różnych technik poszukiwań, a także metody optymalizacji rozwiązań.

Rozdział piąty zawiera główną część pracy. Szczegółowo opisane w nim zostały ograniczenia towarzyszące modelowaniu problemu WDP za pomocą techniki CLP, a także przedstawione zostały dwie metody budowy tych ograniczeń. W rozdziale zaprezentowano technikę tworzenia modeli CLP dla problemu WDP oraz szczegółowo omówiono dwa własne modele CLP (SMO-WDP, DMO-WDP), umożliwiające rozwiązywanie problemu wyznaczania zwycięzcy dla dowolnej aukcji kombinatorycznej.

W rozdziale szóstym przedstawiono szczegóły dotyczące przeprowadzanych testów i analizy prezentowanych w pracy modeli. Scharakteryzowano w nim komputerowe środowisko testowe oraz opisano szczegółowo banchmarkowe dane testowe wraz z ich transformacją. Umieszczono w nim szereg wykresów, będących wynikami przeprowadzonych testów. W rozdziale tym został również szczegółowo opisany każdy z otrzymanych wyników.

W rozdziale siódmym umieszczono podsumowanie niniejszej pracy. Odwołano się do celu pracy oraz tez postawionych na wstępie. Zawarto w nim krótkie podsumowanie wszystkich przeprowadzonych testów i otrzymanych wyników.

Niniejszą pracę kończą trzy dodatki. Pierwszy z nich zawiera praktyczną implementację dwóch przedstawionych w pracy modeli CLP, w internetowym systemie wspomagającym wyznaczenie zwycięzcy aukcji kombinatorycznej. Drugi zawiera spis stosowanych w pracy skrótów i oznaczeń. Trzeci, ostatni zawiera kompletne kody źródłowe prezentowanych i opisywanych w pracy modeli SMO-WDP i DMO-WDP.

Rozdział 2

Aukcje i przetargi kombinatoryczne

Rozdział zawiera podstawowe informacje o aukcjach i przetargach, w tym historię i genezę aukcji. Zawiera również opis klasyfikacji aukcji ze względu na różne ich cechy. Wyjaśnia pojęcie aukcji kombinatorycznej wprowadzając w problem wyznaczenia zwycięzcy aukcji kombinatorycznej.

2.1 Aukcje i przetargi

Aukcja jest jednym z najstarszych sposobów wymiany handlowej. Już w Babilonie około V wieku przed naszą erą aukcje wykorzystywane były do wymiany handlowej [Toc02], głównie niewolników i żon. Aukcje te charakteryzował fakt, że początkowe ceny wywoławcze mogły być ujemne. Obecnie typowy rynek aukcyjny dotyczy zazwyczaj towarów jednostkowych oferowanych w sprzedaży ofertowej jako unikalne i niepodzielne dobra. W tym przypadku wyznaczenie zwycięzcy aukcji, czyli osoby, która wygrywa poprzez złożenie najlepszej oferty (najbardziej opłacalnej dla sprzedajacego) nie stwarza problemu, gdyż aukcje wygrywa ta oferta, która po jej zakończeniu oferuje najwyższą cenę za dany towar. Reguły określające zasady przeprowadzania aukcji mogą być jednak dużo bardziej skomplikowane. Przykładem tego mogą być aukcje kombinatoryczne.

2.2 Klasyfikacja aukcji i przetargów

Ze względu na różne cechy charakteryzujące aukcje i przetargi, możemy określić różne grupy ich klasyfikcji [Toc02], [ND05]. Klasyfikować aucje i przetargi

możemy między innymi pod względem jawności ofert. Wyróżniamy tutaj dwa podstawowe jej typy:

- **Aukcja jawna** - jest aukcją, w której wszystkie zgłoszone na aukcji oferty kupna, zawierające propozycje cenowe, są ogólnie znane wszystkim uczestnikom aukcji. Uczestnicy aukcji jawnej mają pełną informację na temat wszystkich proponowanych cen zawartych w ofertach, jak również posiadają pełną informację na temat liczby uczestników aukcji.

- **Aukcja niejawna** - to aukcja, w której oferenci składają propozycje cenowe dotyczące kupna dóbr w zaklejonych kopartach. W przypadku aukcji niejawnej jej uczestnicy nie znają wartości cen innych składanych ofert, jak również mogą nie posiadać informacji na temat liczby pozostałych uczestników aukcji.

Do najpopularniejszych typów aukcji jawnych możemy zaliczyć dwie grupy, różniące się sposobem wyznaczenia zwycięzcy.

- **Aukcja angielska** - jest aukcją, w której cena, począwszy od niskiej ceny wywoławczej, jest podnoszona do momentu gdy pozostanie tylko jedna osoba licytująca dany towar.

- **Aukcja holenderska** - jest aukcją, w której bardzo wysoka cena rozpoczynająca aukcję jest stopniowo obniżana do momentu, gdy pierwsza z osób licytujących zgłosi chęć kupna licytowanego towaru lub usługi.

Najpopularniejszym typem aukcji niejawnej jest przetarg kopertowy.

- **Przetarg kopertowy** - jest typem aukcji niejawnej, w której oferty są składane w zaklejonych koperatch, dzięki czemu nie są znane innym uczestnikom aukcji. Ponad to uczestnicy aukcji nie posiadają informacji o liczbie złożonych ofert.

Aukcje można klasyfikować również pod względem liczby uczestniczących w niej osób i to zarówno po stronie kupujących jak i po stronie sprzedających.

- **Aukcja jednostronna** - jest najbardziej popularną aukcją w której występuje tylko jeden sprzedający, natomiast może występować wielu kupujących.

- **Aukcja dwustronna** - to aukcja w której może występować wielu kupujących, jak również i wielu sprzedających. Przykładem aukcji dustronnej mogą być giełdy.

Kolejną grupą, według której możemy klasyfikować aukcje, jest złożoność procesu aukcyjnego. Zgodnie z tym podziałem rozróżniamy:

- **Aukcje jednokrotne** - polegają na wyłonieniu oferty lub ofert wygrywających w jednej iteracji aukcyjnej (w jednej rundzie).

- **Aukcje wielokrotne** - mają bardziej złożoną budowę, a wyznaczenie oferty wygrywającej odbywa się w wielu interacjach. Do pierwszego etapu takiej aukcji są dopuszczane wszystkie oferty spełniające jej założenia. Po zakończeniu pierwszego etapu zostają odrzucone oferty nie spełniające warunków kolejnej iteracji. W ten spoób po określonej liczbie iteracji zostają tylko oferty przyjęte.

2.3 Aukcje i przetargi a kombinatoryka

W przypadku aukcji kombinatorycznych oferowane do sprzedaży towary mogą występować w pojedynczych lub wielu egzemplarzach, wielu różnych dóbr. Kupujący mogą składać oferty kupna na dowolne kombinacje tych dóbr i liczbę ich egzemplarzy. Dodatkowo oferowane w aukcji kombinatorycznej dobra mogą być powiązane relacjami i różnymi warunkami, przez co składanie ofert na pewne kombinacje tych dóbr może być niedopuszczalne. Te wszystkie elementy powodują, że wyznaczenie zwycięzcy takiej aukcji kombinatorycznej jest problemem optymalizacji dyskretnej. Z ogólnego punktu widzenia *problem wyznaczenia zwycięzcy* polega na przeglądzie wszystkich możliwych kombinacji ofert i przyjęciu za wygrane te, które maksymalizują przychód z ich sprzedaży. Tego typu podejście ma jednak wadę, która doprowadza do *eksplozji kombinatorycznej*, to znaczy bardzo szybkiego wzrostu liczby możliwych kombinacji ofert ze wzrostem liczby składanych na aukcji ofert oraz sprzedawanych towarów.

2.4 Klasyfikacja aukcji kombinatorycznych a problem WDP

Z perspektywy rozpatrywanego w pracy problemu wyznaczenia zwycięzcy aukcji kombinatorycznej, aukcje możemy klasyfikować na dwa sposoby. Pierwszym z nich jest liczba jednostek oferowanych na aukcji dóbr, drugim jest liczba uczestników aukcji. W przypadku klasyfikacji aukcji kombinatorycznych pod względem liczby jednostek oferowanych dóbr, wyróżniamy dwie podstawowe grupy aukcji kombinatorycznych: *aukcje jednostkowe wielu towarów* oraz *aukcje wielu jednostek wielu towarów*:

- **Aukcja jednostkowa wielu towarów** (ang. single-unit combinatorial auction) – jest aukcją, gdzie proponowane są do sprzedaży pojedyncze jednostki wielu towarów lub usług.

- **Aukcja wielu jednostek wielu towarów** (ang. multi-unit combinatorial auction) – jest aukcją, gdzie proponowane jest do sprzedaży wiele towarów, a każdy z wystawionych towarów występuje w wielu dostępnych jednostkach. Szczególnym przypadkiem aukcji wielu jednostek wielu towarów jest aukcja jednostkowa wielu towarów.

W przypadku klasyfikacji aukcji kombinatorycznych pod względem liczby uczestników biorących w niej udział, wyróżnić możemy *standardowe aukcje jednostronne*, gdzie występuje jeden sprzedawca i wielu kopujących oraz *aukcje dwustronne* gdzie może występować wielu sprzedających i wielu kupujących [XSW05].

W ogólnym przypadku problem wyznaczenia zwycięzcy aukcji kombinatorycznej wygląda następująco. Aukcjoner zgłasza pewien zbiór towarów do sprzedaży oznaczany symbolem: $M = \{M_1, M_2, ..., M_i, ..., M_m\}$, gdzie m jest liczbą tych towarów. Każdy z poszczególnych towarów posiada liczbę dostępnych jednostek oferowanych na aukcji. Liczby te zawierają się w zbiorze $U = \{U_1, U_2, ..., U_i, ..., U_m\}$, gdzie U_i jest liczbą towaru M_i. Wszystkie oferty złożone na aukcji można uważać za elementy zbioru $B = \{B_1, B_2, ..., B_j, ..., B_n\}$. Oferenci w liczbie n, biorący udział w aukcji mogą zgłaszać oferty kupna towarów, będące parami danych $B_j = \langle (a_{1j}, a_{2j}, ..., a_{ij}, ..., a_{mj}), p_j \rangle$, gdzie $a_{ij} \geq 0$ jest liczbą jednostek towaru

i, na które oferent j-ty zgłasza chęć kupna, a $p_j > 0$ jest ceną jaką ten oferent jest skłonny zapłacić za ofertę o specyfikacji $(a_{1j}, a_{2j}, ..., a_{ij}, ..., a_{mj})$.
Problem wyznaczenia zwycięzcy aukcji kombinatorycznej sprowadza się zatem do wyznaczenia ofert wygrywających, to znaczy maksymalizujących dochód z ich sprzedaży. Bardzo często problem ten jest porównywany do problemu alokacji środków (ang. allocation problem) [ND05], gdyż możemy go rozpatrywać jako przydział zasobów (towarów) do uczestników aukcji.

Ze względu na fakt, że problem wyznaczenia zwycięzcy jest problemem kombinatorycznym, do rozważań należy wprowadzić dodatkowe binarne zmienne decyzyjne x_j, które będą przyporządkowane każdej złożonej na aukcji ofercie:

$$x_j \in \{0, 1\} \, \forall j \in \{1, ..., n\}.$$

Sens każdej binarnej zmiennej decyzyjnej jest następujący:
$x_j = 1$ - przyjęcie j - tej oferty w całości,
$x_j = 0$ - odrzucenie j - tej oferty w całości.

Obecnie problem wyznaczenia zwycięzcy aukcji kombinatorycznej można sprowadzić do postaci kanonicznej programowania liniowego dyskretnego.

$$max \sum_{j=1}^{n} p_j x_j \tag{2.1}$$

przy ograniczeniach:

1. dla przypadku aukcji jednostkowej wielu towarów

$$\sum_{j=1}^{n} a_{ij} x_j \leq 1, \forall i = \{1, ..., m\} \tag{2.2}$$

$$x_j \in \{0, 1\}, \forall j \in \{1, ..., n\} \tag{2.3}$$

2. dla przypadku aukcji wielu jednostek wielu towarów:

$$\sum_{j=1}^{n} a_{ij} x_j \leq U_i, \forall i = \{1, ..., m\} \tag{2.4}$$

$$x_j \in \{0,1\}, \forall j \in \{1,...,n\} \tag{2.5}$$

W obu przypadkach aukcji funkcja celu (2.1) maksymalizuje całkowity zysk aukcjonera ze sprzedaży towarów. Ograniczenia (2.2,2.4) mają za zadanie, za pomocą binarnych zmiennych decyzyjnych x_j, zbilansować w ofertach liczby jednostek każdego towaru. Wprowadzone binarne zmienne decyzyjne (2.3,2.5) zapewniają, że każda ze złożonych na aukcji ofert zostanie przyjęta lub odrzucona w całości.

Aukcję kombinatoryczną wielu jednostek wielu twarów, gdzie $U_i = 1, \forall i = \{1,2,...,m\}$ nazywamy jednostkową aukcją kombinatoryczną wielu towarów [SSGL01b].

Jednostkowa aukcja wielu towarów jest szczególnym przypadkiem aukcji wielu jednostek wielu towarów. Jednakże ze względu na istotne różnice w modelowaniu obu typów aukcji za pomocą programowania w logice z ograniczeniami, w dalszej części pracy będą one rozpatrywane oddzielnie.

2.5 Przykład aukcji kombinatorycznych

2.5.1 Aukcja kombinatoryczna jednostkowa wielu towarów

Przykładem aukcji kombinatorycznej jednostkowej wielu towarów może być wyprzedaż fragmentu kolekcji monet. Numizmatyk ze względów finansowych chce sprzedać pięć różnych monet ze swojej kolekcji. Każda z nich występuje jako pojedynczy egzlemplarz. Głównym celem numizmatyka sprzedającego monety jest zmaksymalizowanie zysku z ich sprzedaży. W tym celu ogłoszona została aukcja kombinatoryczna, która pozwala uczestnikom aukcji na składanie ofert zawierających dowolne, możliwe kombinacje z wystawionych na aukcji monet. W celach identyfikacyjnych monety te zostały oznaczone w następujący sposób: M_1, M_2, M_3, M_4, M_5, co zostało przedstawione w tabeli (2.1).
Po określonym czasie trwania aukcji zebrano cztery oferty ich kupna. Oznaczono je w następujący sposób: oferta1 - B_1, oferta2 - B_2, oferta3 - B_3, oferta4 - B_4. Pełne zestawienie zebranych na aukcji ofert przedstawione zostało w tabeli 2.2.

Tabela 2.1: Przykład aukcji jednostkowej wielu towarów - liczby egzemplarzy monet

Monety (M)	M_1	M_2	M_3	M_4	M_5
Liczby sztuk (U)	1	1	1	1	1

Tabela 2.2: Przykład aukcji jednostkowej wielu towarów - zgłoszone oferty kupna

	M_1	M_2	M_3	M_4	M_5	Wartość oferty
B_1	1	-	1	-	-	100
B_2	1	-	-	-	-	150
B_3	-	1	1	-	-	100
B_4	-	1	-	1	1	100

Pytanie brzmi: które oferty należy przyjąć za wygrywające, aby zmaksymalizować dochód ze sprzedaży monet?
W celu wyznaczenia ofert wygrywających należy utworzyć wszystkie możliwe kombinacje ze wszytkich ofert i wybrać te, dla których suma ich wartości cen ofertowych będzie największa. Należy tutaj pamiętać, że każda tworzona kombinacja musi spełniać ograniczenie (2.2).
Rozwiązaniem optymalnym, maksymalizującym zysk ze sprzedaży monet, dla omawianego przykładu jest przyjęcie za wygrywające ofert oznaczonych jako B_2 i B_3, czyli sprzedaż tylko monet oznaczonych jako M_1, M_2, M_3. Łączna, otrzymana w ten sposób kwota wynosi 250. Jak widać w wyniku przeprowadzenia aukcji nie sprzedane zostały monety M_4 i M_5, gdyż ich sprzedaż nie zwiększyłaby kwoty możliwej do uzyskania na aukcji.

2.5.2 Aukcja kombinatoryczna wielu jednostek wielu towarów

Przykładem aukcji wielu jednostek wielu towarów ponownie może być sprzedaż monet przez numizmatyka. Ze względów finansowych pewna osoba chce sprzedać część swojej kolekcji. Żadna z wystawionych do sprzedaży monet nie występuje jednak w pojedynczych egzemplarzach, lecz w liczbie egzemplarzy przedstawionych w tabeli (2.3). Monety te zostały oznaczone

w sposób następujący: M_1, M_2, M_3, M_4, M_5, co zostało przedstawione w tabeli 2.3.

Tabela 2.3: Przykład aukcji wielu jednostek wielu towarów - liczby egzemplarzy monet

Monety (M)	M_1	M_2	M_3	M_4	M_5
Liczby sztuk (U)	6	4	5	2	3

W celu sprzedaży monet ogłoszono aukcję kombinatoryczną. Uczestnicy takiej aukcji mogą składać oferty na dowolne kombinacje wystawionych na aukcji monet. Po upływie określonego czasu trwania aukcji zebrano cztery oferty kupna. Oznaczono je kolejno: oferta1 - B_1, oferta2 - B_2, oferta3 - B_3, oferta4 - B_4. Pełne zestawienie zebranych ofert zostało przedstawione w tabeli (2.4).

Tabela 2.4: Przykład aukcji wielu jednostek wielu towarów - zgłoszone oferty kupna

	M_1	M_2	M_3	M_4	M_5	Wartość oferty
B_1	2	-	3	-	-	100
B_2	4	-	-	-	-	150
B_3	-	2	4	-	-	100
B_4	-	3	-	2	1	100

Podobnie jak w poprzednim przykładzie, w celu wyznaczenia ofert wygrywających należy utworzyć wszystkie możliwe kombinacje zebranych ofert, sumując ich wartości w każdej kombinacji, a następnie dokonać wyboru tych, dla których suma wartości danej kombinacji będzie największa. Należy tutaj pamiętać, że liczba poszczególnych alokowanych towarów we wszystkich ofertach zgodnie z ograniczeniem (2.4) musi być mniejsza lub równa liczbie dostępnych na aukcji jednostek danego towaru.
Rozwiązaniem optymalnym, maksymalizującym zysk ze sprzedaży monet jest przyjęcie za wygrywające następujących ofert - B_1, B_2, B_4. Dla danej kombinacji ofert suma ich wartości wynosi 350. W wyniku przeprowadzenia aukcji nie zostały sprzedane wszystkie monety, ale zysk ze sprzedanych został zmaksymalizowany. Ciekawym elementem tego przykładu jest fakt, że oferta numer

dwa (B_2), niezależnie od wysokości proponowanej ceny, zostałaby i tak przyjęta za wygrywającą w danym przykładzie. Dzieje się tak dlatego, gdyż suma jednostek towaru M_1 we wszystkich złożonych ofertach jest równa wystawionej na aukcji liczbie tego towaru (6 sztuk), a oferta numer dwa (B_2) złożona została tylko w celu zakupu czterech monet M_1.

2.6 Klasyfikacja problemu wyznaczenia zwycięzcy

Problem wyznaczenia zwycięzcy aukcji kombinatorycznej w ogólnym przypadku jest problemem optymalizacji kombinatorycznej. Poszukiwanie rozwiązania tego problemu może teoretycznie odbywać się poprzez *przegląd zupełny kombinacji* wszystkich złożonych ofert i próbę optymalnego dopasowania ich do siebie w ten sposób aby nie naruszały ograniczeń opisanych wcześniej. Tą drogą powstaną kolejne kombinacje, które będą reprezentować pewne rozwiązania pośrednie (dopuszczalne). Podejście takie napotyka jednak bardzo szybko na barierę związaną ze wspomnianą już wcześniej eksplozją kombinatoryczną, która to jest związana z wykładniczą złożonością obliczeniową, charakterystyczną dla większości problemów kombinatorycznych.
Dla przykładu, w wyniku tworzenia wszystkich możliwych kombinacji za pomocą przeglądu zupełnego, dla czterech elementów oznaczonych kolejno 1,2,3,4 możliwych do utworzenia kombinacji jest 15:

```
1
1,2
1,2,3
1,2,3,4
1,2,4
1,3
1,3,4
1,4
2
2,3
2,3,4
2,4
3
3,4
4
```

Jak zauważono model omawianego problemu WDP jest identyczny z modelem problemu plecakowego w programowaniu całkowito-liczbowym, metody programowania całkowito-liczbowego zostały omówione na przykad w [Trz03], [WC78]. Idąc dalej można udowodnić złożoność obliczeniową problemu WDP poprzez sprowadzenie go do ogólnej postaci $NP - trudnego$ problemu plecakowego [LMS06],[Kel04],[Hol01]. Również $NP - trudny$ problem upakowania (ang. Set Packing Problem) jest równoważny z problemem wyznaczenia zwyciezcy [RPH98], [HO04].

Reasumując, problem wyznaczenia zwycięzcy aukcji kombinatorycznej jest w ogólnym przypadku $NP - trudnym$ problemem kombinatorycznym o wykładniczej złożoności obliczeniowej, $O(2^n)$ [VV03],[LMS06]. Oznacza to, że czas potrzebny do rozwiązania problemu rośnie wykładniczo wraz z rozmiarem problemu reprezentowanym przez liczbę złożonych na aukcji ofert (n) tak, że $n \to \infty$.
Szczegółowe informacje na temat złożoności obliczeniowej problemu WDP można znaleźć w niemal każdej pracy poświęconej temu problemowi, a w szczególności w [CSe06].

2.7 Zastosowanie aukcji kombinatorycznych

Aukcje kombinatoryczne mogą mieć swoje zastosowanie wszędzie tam, gdzie mamy do czynienia z ekonomiczną wymianą większej liczby towarów lub usług. Były i są one wykorzystywane między innymi:

- do obrotu na rynkach nieruchomości, a w szczególności do sprzedaży działek obszarowych (ang. Real estate) [Qua94],
- do sprzedaży praw użytkowania radiowych pasm częstotliwościowych (ang. The FCC Spectrum allocation problem) [McM94],
- do obsługi zamówień publicznych (ang. Public procurment) [And03],
- na rynku obrotu ropy naftowej (ang. Crude oil sales)[Sch03],
- na rynku obrotu energii elektrycznej [Toc02],
- do wybranych zagadnień planowania (ang. scheduling) [WWWMM01],

- do obsługi rejestracji kursów (ang. Course registration) [GSS93],
- do modelowania transakcji handlowych (ang. Trading packages) [TWW+01].

Rozdział 3

Obecny stan zagadnienia

W rozdziale tym przedstawiono obecnie znany stan rozwiązywania problemu wyznaczenia zwycięzcy różnych typów aukcji kombinatorycznych. W szczególności opisano znane metody i algorytmy umożliwiające rozwiazywanie problemu.

3.1 Problematyka aukcji

Problematyka modelowania matematycznego aukcji nie jest nowa. W roku 1961 William Spencer Vickrey jako pierwszy opracował model matematyczny i opisał zasady przeprowadzania standardowej aukcji niejawnej pojedynczego obiektu. W swojej pracy [Vic61] zaproponował między innymi model zapobiegający spekulacjom cenowym, polegający na tym, że uczestnik, który złożył najwyższą ofertę, zapłaci za nią cenę, jaką posiadała druga, po wygrywającej, oferta. Aukcja ta obecnie posiada uogólnienia pozwalające na zastosowanie ich w dowolnych aukcjach wielu towarów, w tym także aukcji kombinatorycznych [Cla71].
W kolejnych latach tematyka aukcji była nieco pomijana w badaniach i analizach. Dopiero początkiem lat dziewięćdziesiątych zainteresowano się nią ponownie. Obecnie jest to bardzo dynamicznie rozwijająca się interdyscyplinarna dziedzina naukowa, łącząca w sobie aspekty ekonomii, sztucznej inteligencji, badań operacyjnych, teorii gier oraz metod optymalizacji. ([Trz03])

3.2 Znane metody i algorytmy rozwiązywania problemu WDP

W literaturze przedmiotu przedstawiono wiele metod i algorytmów umożliwiających rozwiązywanie problemu wyznaczenia zwycięzcy aukcji kombinatorycznej. Są to między innymi algorytmy dokładne, takie jak CASS i VSA [FLBS99], CAMUS [LBST00], algorytm Nissana [Nis00], algorytm BOB [SS03], CABOB [SSGL01a] oraz algorytmy przybliżone takie jak stochastyczne przeszukiwanie lokalne [HB00], zmienne przeszukiwanie lokalne (VNS) [FT05]. Można również znaleźć propozycje rozwiązania problemu metodami programowania liniowego (ang. Linear Programming - LP) [GL01] oraz programowania całkowitoliczbowego, (ang. Integer Programming - IP) [Sch03], [ATY00]. Ważniejsze z tych algorytmów można scharakteryzować następująco:

- **CASS (Combinatorial Auction Structured Search)** - algorytm ten służy do rozwiązywania problemu wyznaczenia zwycięzcy jednostkowej aukcji kombinatorycznej. Jest algorytmem dokładnym, opartym o brutalną metodę (naive brute-force) z czterema punktami popraw. Pierwszym z nich jest zastosowanie tak zwanych "koszy"(BINS) do sortowania ofert. Ma to na celu redukcję liczby niewykonalnych przydziałów towarów do ofert. Drugim punktem popraw jest zastosowanie buforowania podręcznego (caching), a kolejnymi dwoma punktami popraw są okrojenia (pruning). Algorytm ten nie jest obecnie wykorzystywany gdyż na jego podstawie został zbudowany inny bardziej efektywny algorytm o nazwie CAMUS.

- **CAMUS (Combinatorial Auction Multi-Unit Search)** - oparty jest o algorytm CASS z zastosowaniem dodatkowej techniki poszukiwań typu branch-and-bound. Algorytm CAMUS polega na systematycznym porównywaniu dochodu wszystkich pełnych przydziałów towarów do ofert. Porównanie to jest implementowane przy pomocy algorytmu przeszukiwania depth-first search (DFS). Jednym z etapów algorytu jest usuwanie z rozważań ofert nie dominujących, to znaczy takich dla których istnieją oferty o takich samych towarach, co w ofertach nie dominujących ale o wyższych cenach.

- **BOB (Branching on bids)** - jest algorytmem dokładnym i polega na przeszukiwaniu drzewa decyzyjnego. Oparty jest o metodę przeszukiwania depth-first branch-and-bound. Algorytm ten nie został zaimplementowany, ale stał się podstawą do powstania bardziej zaawansowanego rozwiązania.

- **CABOB (Combinatorial Auction Branch On Bids)** - oparty jest o algorytm BOB. Działa on na zasadzie przeszukiwania drzewa i gałęzi ofert metodą depth-first branch-and-bound. Do wyznaczenia rozwiązania problemu stosuje dekompozycyjną technikę górnej i dolnej granicy opartą o uporządkowane heurystyki.

- **Zmienne przeszukiwanie lokalne (Variable Neighbourhood Search - VNS)** - algorytm przybliżony oparty o metodę poprawy rozwiązań problemu - metaheurystyki zmiennego przeszukiwania lokalnego [HM01]. W przypadku problemu WDP, w pierwszym kroku oferty kupna są szeregowane zgodnie z funkcją uszeregowania [GL00]. Oferta zostaje przyjęta wtedy, gdy zapotrzebowanie oferty na dany towar nie przekracza jeszcze dostępnych jednostek danego towaru. W przeciwnym wypadku oferta jest odrzucana. W ten sposób powstają kombinacje ofert tworzące rozwiązania dopuszczalne. Rozwiązanie problemu jest reprezentowane przez sekwencje ofert różniących się sąsiedztwem (odpowiadające im sekwencje różnią się tylko pozycją pojedynczej oferty). W kolejnych krokach wykonywane zostają "wstrząsanie"i optymalizacja lokalna w oparciu o zmodyfikowaną metodę najszybszego spadku.

Bardzo dokładne i szczegółowe podsumowanie obecnie znanych metod i opublikowanych algorytmów, umożliwiających rozwiązywanie problemu wyznaczenia zwycięzcy aukcji kombinatorycznej, można znaleźć w [San02]. Praca ta systematyzuje metody i algorytmy służące fo rozwiązywania problemu wyznaczenia zwycięzcy, nie wpomina jednakże o istniejących również kilku opracowaniach poruszających tematykę aukcji kombinatorycznych w kontekście technik programistycznych, zbliżonych do programowania w logice z ograniczeniami.
Pierwszym z tych opracowań jest [BL06], łączy ono w sobie technikę progamowania ograniczeniowego (ang. Constraint Programming - CP) z problemem wyznaczenia zwycięzcy. Opracowanie to posiada charakter bardzo ogólny i jest

pozbawione istotnych szczegółów dotyczących wykorzystania programowania ograniczeniowego do rozwiązywania problemu wyznaczenia zwycięzcy aukcji kombinatorycznych.
Drugim z tych opracowań jest [HO04]. Porusza ono tematykę programowania opartego o środowisko realizujace założenia dla problemu spełnienia ograniczeń, opisanego w dalszej części pracy.
Ostatnim ze znalezionych przez autora opracowań, łączących problem wyznaczenia zwycięzcy z technikami zbliżonymi do programowania w logice z ograniczeniami jest [BU01]. Opracowanie to opisuje semantyczną możliwość zapisu różnych problemów aukcji kombinatorycznych, za pomocą deklaratywnego języka programowania, opartego o środowisko programistyczne o nazwie *Smodel* [NS97]. Środowisko to pozwala na tworzenie modeli zgodnie z konwencją programowania w logice, a nie zgodnie z techniką programowania w logice z ograniczeniami (różnice opisane zostały w kolejnym rozdziale). Zbudowane modele zaprezentowane w pracy umożliwiają rozwiązywanie problemu wyznaczenia zwycięzcy aukcji kombinatorycznych. Autorzy pracy po przeprowadzonych testach opracowanych modeli, otrzymali porównywalne wyniki rozwiązywanych przypadków z obecnie stosowanymi algorytmami imparatywnymi.

Autorowi nie udało się odnaleźć ani jednej pracy rozwiązującej problem wyznaczenia zwycięzcy aukcji kombinatorycznej przy zastosowaniu programowania w logice z ograniczeniami. Nawet w przeglądowej pozycji [RvBW06] nie ma wzmianki o zastosowaniu programowania w logice z ograniczeniami do rozwiązywania problemu wyznaczenia zwycięzcy aukcji kombinatorycznych. Dlatego w niniejszej pracy zdecydowano się na rozwiązanie wybranych problemów optymalizacji aukcji kombinatorycznych od podstaw, z wykorzystaniem techniki *programowania w logice z ograniczeniami*. Problemami tymi są problemy wyznaczenia zwycięzcy aukcji lub przetargów kombinatorycznych.

Rozdział 4

Programowanie w logice z ograniczeniami

Rozdział zawiera informacje ogólne dotyczace metody programowania w logice, a także szczegółowe opisy głównych elementów wchodzących w skład metody programowania w logice z ograniczeniami, w tym problem spełnienia ograniczeń, propagacji ograniczeń oraz optymalizacji rozwiązań. Umieszczono w nim porównanie omawianej metody z tradycyjnymi metodami programowania.

4.1 Programowanie w logice

Programowanie w logice to deklaratywna metoda programowania, w której program jest pewnym zbiorem reguł i faktów, a celem programu jest przeprowadzenie dowodu pewnego twierdzenia opartego o te reguły i fakty.
Pierwszym językiem programowania wykorzystującym metodę programowania w logice (ang. Logic programming) był *Prolog* [GS91]. Powstał on w roku 1972 na Uniwersytecie w Marsylii, a podstawą jego utworzenia była logika predykatów pierwszego rzędu (ang. first order predicate logic).
Program napisany w Prologu zasadniczo różni się od programów napisanych w językach proceduralnych, gdyż nie zawiera on algorytmu rozwiązania lecz jedynie fakty i reguły opisujące problem. Zarówno fakty jak i reguły są klauzulami Horna. Do opisu klauzul stosowane są dowolne ciągi znaków pozbawione znaczenia semantycznego, ogólnie nazywane *termami*. Term może być przedstawiony w postaci stałej, zmiennej lub predykatu - relacji pomiędzy zmiennymi nazwowymi pozbawionej wartości logicznej. Program napisany w języku Prolog jest więc odpowiednim opisem rozwiązywanego problemu.

Prolog znajduje swoje zastosowanie dla prostszych problemów kombinatorycznych oraz w systemach ekspertowych (ang. Expert Systems) opartych o bazy wiedzy [Nie06], a także w różnych dziedzinach związanych z przetwarzaniem symbolicznym [CM03], takich jak:

- relacyjne bazy danych,
- przetwarzanie języka naturalnego,
- rozwiązywanie problemów kombinatorycznych [MB06].

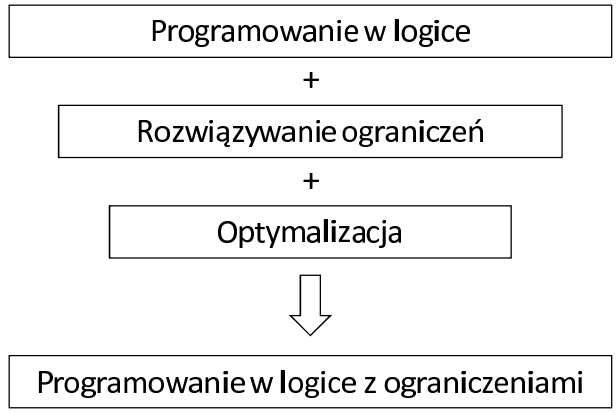
Programowanie w logice stosuje metodę standardowych nawrotów (ang. Standard Backtracking) [Bar98] do przeszukiwania przestrzeni decyzyjnej rozwiazywanego problemu. *Standard backtracking* jest ogólnym algorytmem poszukiwania wartości zmiennych, spełniających pewne ograniczenia. W momencie naruszenia jakiegokolwiek ograniczenia przez wartość ostatnio ukonkretnianej zmiennej mechanizm generowania nawrotów wraca do najbliższej ukonkretnionej zmiennej, dla której istnieje możliwość innego ukonkretnienia. Poszukiwania wartości zmiennych, spełniających wszystkie oraniczenia, odbywają się w drzewie decyzyjnym odpowiadającym przestrzeni decyzyjnej, które jest w trakcie tych poszukiwań dynamicznie tworzone. Jest to jedna z podstawowych właściwości Prologu, umożliwiająca poszukiwania w drzewach o ogromnych rozmiarach, wykluczających ich zapisanie.

Programowanie w logice z zastosowaniem mechanizmu standardowych nawrotów, w porównaniu z tradycyjną metodą przeglądu zupełnego przestrzeni decyzyjnej, pozwala na znaczne ograniczenie przestrzeni decyzyjnej. Ma to istotny wpływ na ograniczenie czasu niezbędnego do wyznaczenia rozwiazań problemu zarówno dopuszczalnych jak i optymalnych.

4.2 Programowanie w logice z ograniczeniami

Programowanie w logice z ograniczeniami zostało opracowane pod koniec lat 80-tych ubiegłego wieku przez J. Jaffara i J.L. Lasseza. Programowanie to ogólnie postrzegać można jako połączenie programowania w logice, rozwiązywania ograniczeń i optymalizacji [JM94]. Podzielić zatem możemy go na dwa główne, w pełni deklaratywne paradygmaty: programowanie w logice i rozwiązywanie ograniczeń.

Zasadniczą różnicą pomiędzy programowaniem w logice a programowaniem w logice z ograniczeniami jest uzupełnienie dziedziny termów dziedziną liczb całkowitych i rzeczywistych oraz uzupelnienie stosowanej w prologu unifikacji działaniami na tych liczbach. W przypadku programowania w logice każdy symbol jest jedynie łańcuchem znaków, który nie jest w żaden sposób interpretowany, a jedynie porównywany leksykograficznie z innymi symbolami. Programowanie w logice z ograniczeniami pozwala wprowadzić dodakowe znaczenie dla każdego symbolu związane z domeną odpowiadającą danemu symbolowi.



Rysunek 4.1: Programowanie w logice z ograniczeniami

Programowanie w logice z ograniczeniami ułatwia rozwiązywanie wielu problemów, w tym i kombinatorycznych problemów decyzyjnych, ponieważ modelowanie problemów nie wymaga od programisty tworzenia skomplikowanego algorytmu rozwiązania problemu (tworzenia metody poszukiwania rozwiązania w drzewie decyzyjnym), a jedynie szczegółowy jego opis. W ten sposób pomiędzy opisem problemu, a programem (modelem) jego rozwiązania nie ma przepaści semantycznej, co znacznie wpływa na czytelność programu.

Programowanie to z powodzeniem można stosować do rozwiązywania następujących problemów:

- kombinatoryczne problemy przeszukiwania,
- optymalizacja dyskretna,
- planowanie,

- szeregowanie,
- przydział zasobów.

4.2.1 Rozwiązwywanie ograniczeń

Rozwiązywanie ograniczeń w programowaniu w logice z ograniczeniami jest połączone z dobrze znanym problemem spełnienia ograniczeń (ang. Constraint Satisfaction Problem - CSP) [Apt03]. Problem ten jest tematem badań w dziedzinie sztucznej inteligencji od wielu lat [Bar99]. Obliczeniowe problemy towarzyszące różnym dziedzinom badań mogą być modelowane i rozwiązywane jako problemy z zakresu spełnienia ograniczeń. CSP opisuje założenia skończonej liczby zmiennych i skończonej liczby wartości dla każdej z tych zmiennych. Problem spełnienia ograniczeń można zatem zdefiniować następująco:

- zbiór zmiennych całkowitych $X = \{x_1, ..., x_n\}$
- dla każdej zmiennej całkowitej x_i skończony zbiór D_i nazywany domeną
- zbiór ograniczeń - będący podzbiorem niedopuszczalnych elementów Iloczynu Kartezjańskiego $D_1 x D_2 x ... x D_n$

Rozwiązaniem dopuszczalnym problemu spełnienia ograniczeń jest każdy ukonkretniony zbiór $X = \{x_1, ..., x_n\}$, nie będący elementem podzbioru niedopuszczalnych elementów Iloczynu Kartzjańskiego $D_1 x D_2 x ... x D_n$.
Rozwiązaniem optymalnym problemu spełnienia ograniczeń nazywa się takie rozwiązanie dopuszczalne, które maksymalizuje lub minimalizuje określony wskaźnik jakości.
Problem spełnienia ograniczeń dotyczy wszystkich ograniczeń i polega na przeglądzie przestrzeni możliwych rozwiązań i dopasowaniu dopuszczalnych wartości do zmiennych przy spełnieniu wszystkich ograniczeń. Ograniczenia modelowanego problemu decyzyjnego określają jakie wartości są dopuszczalne i dla których zbiorów zmiennych. Mogą one być definiowane wprost poprzez ustawienie dopuszczalnych wartości dla zmiennych lub ukrycie przy pomocy algebraicznych relacji. Teoretycznie problem spełnienia ograniczeń można zrealizować tworząc prosty algorytm, który za pomocą przeglądu zupełnego metodą generowania kombinacji i sprawdzania spełnienia ograniczeń przegląda przestrzeń rozwiązań. Jednakże w przypadku rzeczywistych problemów decyzyjnych przestrzeń decyzyjna, którą algorytm musiałby sprawdzić okazuje się

zbyt duża, a czas potrzebny na sprawdzenie wszystkich możliwych do utworzenia kombinacji przez algorytm zbyt długi.

Dlatego w celu zwiększenia efektywności przeglądania przestrzeni decyzyjnej zastosować można programowanie w logice z ograniczeniami, wykorzystujące do przeglądu przestrzeni decyzyjnej metodę standardowych nawrotów i ich modyfikacje, połączoną z propagacją ograniczeń. Pozwala to na znaczne ograniczenie zbioru poszukiwań, a także umożliwia wyznaczenie wszystkich rozwiązań dopuszczalnych, zgodnych z problemem spełnienia ograniczeń.
Metoda programowania w logice z ograniczeniami podczas rozwiązywania ograniczeń pozwala wykorzystać tak zwane **aktywne ograniczenia**. Są to ograniczenia mogące zawierać zmienne nie ukonkretnione, a więc pozbawione wartości. Aktywacja takiego ograniczenia równoważna jest z poszukiwaniem ukonkretnienia dla tych zmiennych.

4.2.2 Propagacja ograniczeń

Większość problemów decyzyjnych z zakresu optymalizacji kombinatorycznej to problemy *NP-zupełne*, [Har01] a więc praktycznie nie rozwiązywalne dla dużej liczby zmiennych. Dlatego przeszukiwanie przestrzeni rozwiązań problemów decyzyjnych przy pomocy przeglądu zupełnego, bez wykorzystania dodatkowych metod ograniczania tego zbioru, jest nie efektywne. Jedną z metod ograniczania przestrzeni decyzyjnej w programowaniu w logice z ograniczeniami jest zastosowanie propagacji ograniczeń. Istotą propagacji ograniczeń jest stosowanie ograniczeń dla redukcji domen zmiennych. Uwzględnienie nowego ograniczenia skutkuje automatyczną modyfikacją domeny wszystkich zmiennych, związanych z tym ograniczeniem. Założeniem propagacji ograniczeń jest automatyczne usuwanie z domeny wszystkich tych zmiennych, które nie spełniają propagowanych ograniczeń. Każde deklarowane w modelu ograniczenie propagowane jest osobno, przy czym każda propagacja ograniczenia za każdym razem akualizuje wartości wszystkich pozostałych domen.
Przykładem problemu spełniania ograniczeń z wykorzystaniem propagacji ograniczeń mogą być zmienne ze skończonych domen x, y, z gdzie:

$$x \in \{1..10\}, y \in \{1..10\}, z \in \{1..10\}$$

przy ograniczeniach:

$$y < z, x = y + z, x = z + 3$$

W wyniku działania propagacji ograniczeń domeny zmiennych zostały zmniejszone do postaci:

$$x \in \{5..10\}, y \in \{1..6\}, z \in \{2..7\}$$

Istnieje wiele różnych metod propagacji ograniczeń [Bar99]. Do najbardziej popularnych można zaliczyć: *Forward Checking (FC)* i *Looking Ahead (LA)* oraz połączenie obu technik *Looking Ahead + Forward Checking FC+LA*. W przypadku FC po ukonkretnieniu nowej zmiennej, z domen pozostałych zmiennych są usuwane wartości niezgodne z już ukonkretnionymi zmiennymi. Mechanizm nawrotu (backtraking) jest tutaj inicjowany w wyniku pojawienia się domeny pustej. FC+LA po ukonkretnieniu nowej zmiennej, z domen pozostałych zmiennych usuwane są wartości sprzeczne z już ukonkretnionymi zmiennymi (Forward Checking). W FC+LA dodatkowo sprawdza się, czy domeny zmiennych nie ukonkretnionych mają wartości dopuszczalne (Looking Ahead). Nawrót jest inicjowany w wyniku pojawienia się domeny pustej lub domen niepustych, lecz pozbawionych wartości dopuszczalnych. Zdarzeniem wyzwalającym nawrót nie jest więc naruszenie ograniczenia lecz nieuchronność naruszenia ograniczenia w następnym kroku (Forward Checking) oraz nieuchronność naruszenia ograniczenia w dalszych krokach (Looking Ahead).

4.2.3 Strategie numeracyjne

Ze względu na fakt, że sama propagacja ograniczeń nie doprowadzi do wyznaczenia rozwiazań dopuszczalnych i rozwiązania optymalnego, a jedynie do zmniejszenia rozmiarów domen (zmniejszenia rozmiaru problemu), konieczne jest zastosowanie metody przeszukiwania pozostałych domen (gałęzi drzewa decyzyjnego problemu). Podstawowym mechanizmem przeszukiwania gałęzi drzew decyzyjnych w programowaniu w logice jest mechanizm standardowych nawrotów. Programowanie w logice z ograniczeniami pozwala dodatkowo na wykorzystanie różnych *strategii numeracyjnych*.

Strategie numeracyjne pozwalają na częściowe sterowanie procesem poszukiwań w drzewie decyzyjnym. Można tutaj mówić o częściowym procesie sterowania poszukiwaniami, gdyż strategie numeracyjne związane są z kolejnością ukonkretniania zmiennych, a zatem ze zmianą metody poszukiwania

optimum globalnego. Obecnie większość solwerów CLP posiada wbudowane mechanizmy pozwalające na wybór strategii numeracyjnych, a także metody ich zmiany.
Wybór typu strategii numeracyjnej dla dowolnego rozwiązywanego problemu w pierwszym etapie zazwyczaj odbywa się na zasadzie wybierz strategię i ją testuj, a następnie wybierz inną i ponownie ją testuj. Dopiero po przeprowadzeniu odpowiednich testów wszystkich możliwych typów strategii numeracyjnych dla danego modelu, można jednoznacznie określić, która strategia jest dla danego problemu (modelu) najlepsza.
Aby skorzystać z możliwości zastosowania strategii numeracyjnych podczas ukonkretniania zmiennych za pomocą solwera CHIP v5.8 [Con], należy zastosować w modelu rozwiązującym dany problem czteroargumentowy predykat *labeling*(_, _, _, _). Wyróżniamy następujące strategie numeracyjne możliwe do wykorzystania, a wbudowane w ten solwer:

- **first fail** - wybierana jest domena, która posiada najmniejszą liczbę elementów. Strategia ta używana jest głównie wtedy, gdy liczba ograniczeń jest taka sama dla wszystkich zmiennych,

- **most constrained** - wybierana jest ta domena, która ma najmniejszą liczbę elementów, w praktyce metoda ta działa bardzo dobrze dla większości przypadków,

- **smallest** - wybierana jest domena zmiennych, która posiada najmniejszą wartość tych domen. Heurystyka ta jest użyteczna podczas rozwiązywania problemów upakowania, gdzie dostępna przestrzeń wartości przydzielanych powinna być upakowana możliwie gęsto,

- **largest** - wybierana jest domena zmiennych, która posiada największą wartość tych domen. Jest heurystyką przeciwną do *smallest* i może być stosowana alternatywnie, jeżeli górna granica domeny jest znana,

- **max regret** - wybierane jest ograniczenie tej domeny, która ma największą różnicę wartości pomiędzy najmniejszą i następną po najmniejszej wartości zmiennych w tej domenie.

4.2.4 Optymalizacja rozwiązań

Szeroko stosowanym algorytmem, służącącym do wyznaczania rozwiązania optymalnego, jest algorytm o nazwie *„Branch and Bound"*, ogólnie rzecz biorąc algorytmy z dziedziny *„Branch and Bound"* mają na celu znalezienie rozwiązania optymalnego w oparciu o wartość wskaźnika jakości. Umożliwia on uzyskanie optimum globalnego dla nieliniowego wskaźnika jakości przy ograniczeniach nieliniowych. Ogólna koncepcja algorytmu *„Branch and Bound"* polega na znalezieniu dolnego ograniczenia wartości wskaźnika jakości i odrzuceniu wszystkich tych gałęzi drzewa rozwiązań, które dają wartości gorsze od aktualnego dolnego ograniczenia, a następnie aktualizacji dolnego ograniczenia w przypadku znalezienia rowiązania lepszego od aktualnie stosowanego dolnego ograniczenia. W tym celu mogą wykorzystywać jedną z dwóch metod przeglądu drzewa decyzyjnego: przeszukiwanie w głąb drzewa decyzyjnego (ang. Depth First Search) - lub przeszukiwanie w szerz (ang. Breadth First Search). Metoda przeszukiwania drzewa decyzyjnego w głąb, ze względu na swoje właściwości, jest metodą częściej stosowaną.

"Branch and bound" w przypadku programowania w logice z ograniczeniami do przeglądu przestrzeni rozwiązań, rozwiązywanych problemów wykorzystuje udoskonaloną w *propagację ograniczeń* (Forward Checking + Looking Ahead) wersję standardowego backtrackingu [Nie01].
Zastosowanie algorytmu *„Branch and Bound"* w programowaniu w logice z ograniczeniami, wraz z mechanizmem standardowych nawrotów, wzbogaconego o propagację ograniczeń, pozwala na znajdywanie rozwiązań coraz bliższych rozwiązaniu optymalnemu, a przy okazji na redukcję domen zmiennych w trakcie działania programu. Interpreter języka CLP dodając ograniczenia do zbioru ograniczeń dokonuje propagacji tych ograniczeń aby zapewnić spójność tego zbioru, a to przeważnie prowadzi do jego ograniczania. Formy spójności jakie są zazwyczaj sprawdzane to spójność łukowa (ang. arc consistency), spójność ponadłukowa (ang. hyper-arc consistency) oraz spójność krańców przedziałów (ang. bound consistency). Dodatkowo podczas propagowania kolejnego ograniczenia sprawdzany jest wskaźnik jakości generowanego rozwiązania. W przypadku otrzymania przez ten wskaźnik wartości gorszej od obecnie optymalnej, generowany jest nawrót, co pozwala na dodatkowe, jeszcze większe ograniczenie zbioru poszukiwań przestrzeni

decyzyjnej.

W przypadku wielkich drzew decyzyjnych, a takie występują w problemie WDP, algorytm *„Branch and Bound"* napotyka na barierę NP-zupełności. Czas poszukiwania rozwiązania optymalnego w takim przypadku może się nieskończenie wydłużać. Dlatego programowanie w logice z ograniczeniami stosuje ten algorytm w połączeniu z heurystykami pozwalającymi na obcinanie ogromnych obszarów drzewa poszukiwań, co wpływa na skrócenie czasu generowania rozwiązania optymalnego.
Algorytm imparatywny, umożliwiający rozwiązywanie problemu WDP, oparty o koncepcję *„Branch and Bound"* i metodę programowania liniowego został zaproponowany w [SY04].

4.2.5 CLP a tradycyjne metody programowania

Metoda programowania w logice z ograniczeniami staje się coraz bardziej popularną metodą rozwiązywania problemów kombinatorycznych. Istnieje w literaturze wiele rożnych opracowań, wskazujących na istotne różnice zawarte pomiędzy programowaniem w logice z ograniczeniami a tradycyjnymi metodami programowania [Han03], [Szc02]:

1. Podstawową różnicą pomiędzy CLP a tradycyjnymi metodami programowania jest fakt, iż programowanie w logice z ograniczeniami jest programowaniem deklaratywnym. Nie wymaga ono formułowania algorytmu rozwiązania problemu, co eliminuje przepaść semantyczną pomiędzy problemem a programem go rozwiązującym. Tradycyjne języki programowania takie jak C++, Pascal itd. są natomiast oparte o programy proceduralne (imparatywne) wymagające specyficznych niejednokrotnie bardzo zawiłych algorytmów rozwiązujących problem. Dodatkowo w tradycyjnym podejściu można łatwo zauważyć daleko idące różnice semantyczne pomiędzy pierwotnym sformułowaniem problemu a problemem transformowanym do postaci programu proceduralnego.

2. Oprogramowanie CHIP V5.8 wykorzystywane do tworzenia aplikacji w opraciu o metodę programowania w logice z ograniczeniami posiada wbudowany solwer dla całkowito-liczbowego oraz ciągłego programowania liniowego. Użycie tego solwera nie wiąże się ze sprowadzaniem problemów do określonych postaci standardowych. Solwer interpretuje model

problemu takim, jak został on sformułowany. W przypadku stosowania tradycyjnych solwerów większość z nich wymaga transformacji problemu do postaci akceptowalnej przez program.

3. W przypadku programowania w logice z ograniczeniami umieszczenie dodatkowych ograniczeń w programie nie wpływa na pogorszenie czytelności programu (modelu), natomiast ma istotny wpływ na ograniczenie czasu potrzebnego do znalezienia rozwiązania problemu. W przypadku tradycyjnych metod programowania implementacja dodatkowych ograniczeń w kod programu jest w większości przypadków trudnością, która również bezpośrednio wpływa na zmniejszenie czytelności programu.

4.2.6 Narzędzia CLP

Obecnie istnieje wiele różnych narzędzi programistycznych, umożliwiających tworzenie modeli i programów zgodnie z konwencją programowania w logice z ograniczeniami. Większość z tych narzędzi wywodzi się bezpośrednio z *Prologu* i częściowo bazują one na rozwiązaniach zaimplementowanych w tym pionierskim rozwiązaniu. Do najpopularniejszych z nich można zaliczyć między innymi:

- **CLP(R)** - język opracowany przez IBM, celem przyświecającym jego powstaniu było umożliwienie rozwiązywania problemów z liniowymi ograniczeniami w dziedzinie liczb rzeczywistych (równania i nierówności) [Mon].
- **PROLOG III** - język opracowany w 1990 roku we Francji przez Alaina Colmerauera [Col]. Opiera sie na liniowej arytmetyce wymiernej i zmiennych boolowskich oraz łańcuchach znaków i listach.
- **CHIP** - jest jednym z najbardziej znanych kompletnych środowisk programistycznych, jest produktem komercyjnym firmy COSYTEC [Con]. Udostępnia narzędzia do rozwiązywania zadań z liniowymi ograniczeniami w takich dziedzinach jak: liczby wymierne oraz boolowskie.
- **GNU Prolog** - to darmowa implementacja CLP stworzona przez Daniela Diaza [Dia]. Jest implementacją bardzo popularną ze względu na dostępność wersji na różne platformy między innymi Windows, Linux, SunOS. Gnu Prolog umożliwia stosowanie ograniczeń w dziedzinach skończonych.

- **SWI-Prolog** - jest implementacją Prologu opartą na idei open source, rozwijaną od 1987 roku [Wie]. Autorem jest Jan Wielemaker. Posiada bogaty zestaw bibliotek i narzędzi, implementacja ta znajduje zastosowanie głównie w celach edukacyjnych. SWI-Prolog może być instalowany na różnych platformach, w tym: Unix, Windows, MS-DOS, Sun, Macintosh.

- **ECL^{i}PS^e** - jest systemem implementującym CLP, bazującym na Prologu. Głównym celem powstania ECL^{i}PS^e były potrzeby planowania, zarządzania siłą roboczą, wyznaczanie tras czy alokacja zasobów. Obecnie jest rozwijany jako produkt open source na zasadzie Mozilla Public License 1.1 [Sys]. Jego główną cechą jest zastosowanie ograniczeń w dziedzinach: całkowitoliczbowej, boolowskiej, symbolicznej, liczb wymiernych i rzeczywistych.

Wszystkie opisane wcześniej języki i narzędzia programowania opartego o technikę programowania w logice z ograniczeniami, umożliwiają wyznaczenie rozwiązań dopuszczalnych i optymalnych dla problemów całkowitoliczbowych. Niektóre z nich umożliwiają ponadto rozwiązywanie zadań programowania liniowego, nie wymagając doprowadzenia problemu do postaci kanonicznej.

Rozdział 5

Modelowanie problemu WDP za pomocą CLP

W rozdziale tym szczegółowo omówione zostały metody budowy dwóch modeli CLP dla problemu WDP. W tym reprezentacja i zapis danych wejściowych, podział i metody budowy ograniczeń oraz metody modelowania funkcji celu. W końcowej części tego rozdziału zaprezentowane zostały kody źródłowe tych modeli.

5.1 Budowa modelu CLP dla problemu WDP

Modele CLP, umożliwiające rozwiązywanie problemu WDP, posiadają typową budowę, charakteryzującą większość modeli napisanych za pomocą techniki CLP. Ogólnie składają się one z następujących elementów składowych:

- definicja domen zmiennych,
- definicja ograniczeń,
- sformułowanie celu problemu.

Jak widać, w powyższych elementach nie występuje algorytm rozwiązania problemu, tak jak to ma miejsce w tradycyjnym programowaniu strukturalnym. Jak już wcześniej wspomniano, technika CLP nie wymaga sformułowania przez użytkownika algorytmu rozwiązania problemu, a jedynie odpowiedniego sformułowania opisu tego problemu. Pod *odpowiednio sformułowanym opisem problemu* rozumie się taki opis, który jest zrozumiały dla stosowanego kompilatora języka CLP.

5.2 Podział i definicja domen zmiennych

Prawie każdy program komputerowy działa lub może działać w oparciu o pewne dane, nazywane danymi wejściowymi. Dane te mogą być wewnętrznie ukryte w programie lub też mogą być pobierane podczas działania programu ze źródeł zewnętrznych, na przykład z plików tekstowych. Zależy to zarówno od charakteru i funkcjonalności danego programu jak i od jego przeznaczenia. Zadaniem porogramu w takim przypadku jest odpowiednie przetworzenie tych danych oraz wygenerowanie na ich podstawie wyniku. Format, jaki mogą przyjmować dane wejściowe w programach, zależy od metody programowania oraz funkcjonalności wykorzystywanego kompilatora, a także od umiejętności programisty. Pierwszym etapem powiązania danych wejściowych z modelem CLP jest *definiowanie ich nazw oraz ich domen.* W przypadku programowania techniką CLP, możliwa jest deklaracja danych wejściowych na wiele różnych sposobów. Najbardziej efektywnym sposobem modelowania danych wejściowych dla problemu WDP, z punktu widzenia deklaracji danych oraz ich późniejszego przetwarzania przy zastosowaniu techniki CLP jest zastosowanie elastycznych, uporządkowanych struktur danych, a więc list. Elementami list mogą być w zależności od potrzeb: stałe, zmienne oraz inne listy. Każdą listę w porgramowaniu w logice z ograniczeniami rozpoczyna znak [natomiast kończy]. Elementy listy są odzielane przecinkami, dla przykładu, zapis listy zawierającej trzy elementy 1, 1, 2 będzie wyglądał następująco: $[1, 1, 2]$.

Aukcje kombinatoryczne, podobnie jak większość problemów decyzyjnych, posiadają pewne charakteryzujące je zbiory danych wejściowych. Zbiory te można opisać w następujący sposób:

- *oferty* - zbiór danych, zawierający specyfikację wszystkich złożonych na aukcji ofert,
- *ceny* - zbiór danych, zawierający wszystkie wartości cen dla poszczególnych specyfikacji (złożonych ofert),
- *towary* - zbiór danych, zawierający liczby jednostek wszystkich towarów wystawionych na aukcji.

Wszystkie dane, zawierające informacje o towarach i ofertach, powinny być tak modelowane, aby ułatwiały formalny ich zapis, a także późniejsze łatwe

przetwarzanie tych danych w programie. Inaczej mówiąc nie powinny one komplikować opisu ograniczeń i funkcji celu, a także wyników końcowych. Dokładny opis i charakterystykę danych wejściowych można znaleźć w prawie każdej pracy poświęconej aukcjom kombinatorycznym, między innymi w [GSL06].

Po przeprowadzeniu testów, dla wielu różnych możliwych metod deklaracji danych wejściowych w modelach CLP, wybrany został najbardziej efektywny sposób ich reprezentacji, a mianowicie lista list, czyli uporządkowana struktura danych, będąca charakterystycznym elementem dla programowania w logice z ograniczeniami. Wybrany sposób reprezentacji danych ułatwia zarówno sposób samej deklaracji tych danych, jak i późniejszego ich przetwarzania. Przykład deklaracji danych dla specyfikacji ofert zapisany za pomocą listy list wygląda następująco:

$$
\begin{aligned}
&specyfikacje([\\
&\quad [a_{1,1}, a_{2,1}, \ldots, a_{i,1}, \ldots, a_{n,1}],\\
&\quad [a_{1,2}, a_{2,2}, \ldots, a_{i,2}, \ldots, a_{n,2}],\\
&\quad [\ldots, \ldots, \ldots, \ldots, \ldots, \ldots],\\
&\quad [a_{1,j}, a_{2,j}, \ldots, a_{i,j}, \ldots, a_{n,j}],\\
&\quad [\ldots, \ldots, \ldots, \ldots, \ldots, \ldots],\\
&\quad [a_{1,m}, a_{2,m}, \ldots, a_{i,m}, \ldots, a_{n,m}]\\
&\quad]).
\end{aligned}
$$

Fragment kodu źródłowego modelu CLP dla problemu WDP aukcji kobinatorycznej jednostkowej wielu towarów, zawierający deklarację specyfikacji ofert dla przykładu 2.1 wygląda następująco:

```
specyfikacje([
              [1,1,0,0],
              [0,0,1,1],
              [1,0,1,0],
              [0,0,0,1],
              [0,0,0,1]
              ]).
```

Kolejne wiersze powyższego zapisu reprezentują poszczególne towary, natomiast kolumny reprezentują poszczególne oferty złożone na aukcji. Zero jest interpretowane jako brak danego towaru w ofercie, natomiast jedynka odpowiada umieszczeniu towaru w ofercie.
Taka deklaracja specyfikacji ofert jest uniwersalna dla obu typów aukcji kombinatorycznych (jednostkowych wielu towarów i wielu jednostek wielu towarów). W przypadku aukcji jednostkowej umieszczenie towaru w ofercie jest reprezentowane przez liczbę 1, natomiast w przypadku aukcji wielu jednostek wielu towarów liczba 1 jest zastępowane liczbą jednostek danego towaru, którą uczestnik aukcji chce kupić. Dla przykładu aukcji wielu jednostek wielu towarów 2.3 deklaracja specyfikacji ofert w modelu CLP wygląda następująco:

```
specyfikacje([
              [2,4,0,0],
              [0,0,2,3],
              [3,0,4,0],
              [0,0,0,2],
              [0,0,0,1]
              ]).
```

Kolejnym zbiorem danych wejściowych, niezbędnym do rozwiązania problemu WDP w przypadku aukcji kombinatorycznych, są *ceny* (wartości cen poszczególnych specyfikacji ofert). W przypadku programowania za pomocą metody CLP, *ceny* to pojedyncza lista zawierająca ceny poszczególnych ofert. Dla obu przykładów 2.1 i 2.3, reprezentujących dwa typy aukcji kombinatorycznych, deklaracja cen wszystkich ofert wygląda tak samo.

```
ceny([100,150,100,100]).
```

Kolejnym zbiorem danych wejściowych są *towary*, a w zasadzie wystawione na aukcji liczby jednostek tych towarów. Podobnie jak w przypadku wartości cen tak i tutaj, deklaracją danych wejściowych jest pojedyncza lista. Dla przypadku aukcji jednostkowej wielu towarów jest to lista jednostkowa zawierająca tyle wartości jedności, ile zostało wystawionych na aukcji towarów, na przykład:

```
jednostki([1,1,1,1,1]).
```

Dla przypadku aukcji wielu jednostek wielu towarów jedynki zostają zamienione liczbami fizycznie dostępnych na aukcji jednostek poszczególnych towarów, na przykład:

```
jednostki([6,4,5,2,3]).
```

Przedstawiony zapis wszystkich danych wejściowych problemu WDP dla oprogramowania CHIP v5.8 ma dwa ograniczenia. Pierwszym z nich jest fakt, że wszystkie zapisywane w modelu wartości muszą być liczbami całkowitymi. Drugim ograniczeniem jest rozmiar domen. Oprogramowanie CHIP v5.8 umożliwia tworzenie domen zmiennych z maksymalną wartością dla domeny równą 100000. W ten sposób ograniczona zostaje domena *ceny*, co bezpośrednio ma wływ na ograniczenie wartości zmiennych tej domeny. W przypadku gdy suma wartości cen wszystkich złożonych na aukcji ofert jest większa od 100000, wartości tych ofert muszą zostać odpowiednio przeskalowane. To ograniczenie ma istotny wpływ na rozwiązanie problemu WDP w przypadku, gdy istnieją bardzo duże rozbieżności w wartościach cen poszczególnych ofert.

Oferenci, w zależności od typu i reguł aukcji, mogą czasem dodatkowo zgłaszać swoje preferencje dotyczące ofert. Preferencje te mogą określać, czy oferty złożone przez tego samego oferenta mają być względem siebie *komplementarne* czy też *substytucyjne*. Modelowanie tego faktu zostało przedstawione w punkcie 5.4, dotyczącym modelowania i parametryzacji ograniczeń.

5.3 Podział ograniczeń

Każdy problem decyzyjny posiada pewne ograniczenia. W przypadku problemu WDP występują różne typy ograniczeń. Podstawową grupą tych ograniczeń są ograniczenia ogólne, które zawsze muszą zostać spełnione. Do grupy tej można zaliczyć ograniczenie ofert wykluczających się, to znaczy takich, które zawierają te same towary lub usługi. Przyjęcie jednej z ofert, zawierającej dany towar, skutkuje tym, że nie może zostać przyjęta żadna inna oferta, zawierająca ten sam towar, o ile liczba dostępnych jednostek danego towaru równa jest jeden (aukcja jednostkowa wielu towarów). W przypadku aukcji wielu jednostek wielu towarów ograniczenie to mówi, że liczba jednostek danego towaru zadeklarowanego w ofertach określonych mianem wygrywające musi być mniejsza lub równa liczbie jednostek wystawionych na aukcji.

Kolejnym ograniczeniem, należącym do grupy ogólnych jest ograniczenie mówiące o niepodzielności złożonych na aukcji ofert. Mówi ono, że złożone na

aukcji oferty muszą zostać przyjęte lub odrzucone w całości, to znaczy, że nie dopuszcza się ich podziału.

Dodatkowe ograniczenia, które mogą występować podczas rozwiązywania problemu WDP, są ograniczeniami związanymi z substytucyjnością i komplementarnością składanych ofert. W zależności od reguł aukcji zakłada się zezwolenie na wyrażanie własnych preferencji lub potrzeb za pomocą substytucyjności i komplementarności składanych ofert:

- *Ograniczenie substytucyjne* mówi o tym, że osoba biorąca udział w aukcji i składająca pewne oferty kupna może zastrzec chęć kupna tylko jednej ze złożonych ofert. Przykładem tego może być zgłoszenie na aukcji dwóch ofert z zastrzeżeniem, że jesteśmy zainteresowani kupnem dowolnej, ale tylko jednej z nich.
- *Ograniczenie komplementarne* mówi o tym, że osoba biorąca udział w aukcji i składająca pewne oferty kupna może zastrzec chęć kupna wszystkich towarów, które występują w zgłoszonych ofertach. W ten sposób wszystkie złożone oferty objęte ograniczeniem komplementarnym muszą zostać przyjęte lub odrzucone w całości.

5.3.1 Ograniczenia ogólne

Podstawowym ograniczeniem, które każda oferta brana pod uwagę podczas rozwiązywania problemu wyznaczania zwycięzcy musi spełniać, jest ograniczenie związane z wykluczaniem się ofert. Ograniczenie to należy rozpatrywać osobno dla aukcji jednostkowych wielu towarów oraz dla aukcji wielu jednostek wielu towarów.
W przypadku aukcji jednostowych ograniczenie to mówi, że dany, towar znajdujący się w ofercie przyjętej za wygrywającą, nie może się znaleźć w żadnej innej ofercie w zbiorze ofert wygrywających. Inaczej mówiąc każdy z towarów wystawionych na aukcji może się znaleźć tylko w jednej z ofert przyjętych za wygrywające.

$$\sum_{j=1}^{n} a_{ij} x_j \leq 1, \forall i = 1, 2, ..., m, \tag{5.1}$$

Ograniczenie ofert wykluczających się dla przypadku aukcji wielu jednostek mówi, że suma jednostek danego towaru we wszystkich ofertach przyjętych za

wygrywające musi być mniejsza lub równa liczbie jednostek wystawionych na aukcji.

$$\sum_{j=1}^{n} a_{ij} x_j \leq u_i, \forall i = 1, 2, ..., m, \tag{5.2}$$

5.3.2 Ograniczenia substytucyjne

Poprzez zgłaszane na aukcji ograniczenia substytucyjne, uczestnicy aukcji mogą wyrazić własne preferencje dotyczące składanych ofert, a także towarów wystawionych na aukcji. Odpowiedź na pytanie, czy na danej aukcji można składać oferty z zastrzeżeniem ich substytucyjności, powinna być zapisana w regułach danej aukcji i znana przed jej rozpoczęciem. Ograniczenia substytucyjne mogą dotyczyć zarówno towarów jak i ofert.

O substucyjności *towarów* możemy mówić tylko i wyłącznie w przypadku aukcji kombinatorycznych jednostkowych wielu towarów. Oferent składając więcej niż jedną ofertę może zaznaczyć, że oferty te mają charakter subsytytucyjny. Oznacza to, że tak naprawdę jest zainteresowany kupnem tylko jednej ze złożonych ofert. Ponieważ omawiana aukcja jest jednostkową wielu towarów, możemy wnioskować, że oferent pomimo złożenia wielu ofert, jest tak na prawdę zainteresowany tylko jednym towarem. Stąd bierze się fakt określenia substytucyjności towarów.

Substytucyjność *ofert* ma miejsce w przypadku aukcji wielu jednostek wielu towarów. Ponieważ oferent może składać oferty na zbiory towarów (zawierające więcej niż jeden towar), a nie na pojedyncze towary, substytucyjność w tym przypadku dotyczy ofert. Oferent zaznaczając, że składa na aukcji oferty substytucyjne informuje w ten sposób aukcjonera, że pomimo złożenia wielu ofert jest zainteresowany kupnem tylko jednej z nich (dowolnej).
W przypadku składania wielu ofert z zastrzeżeniem ich substytucyjności fakt ten musi zostać dodatkowo ujęty w modelu. Aby model umożliwiał wyznaczenie rozwiązania dla dowolnej aukcji kombinatorycznej zgodnie ze złożonymi ofertami substytucyjnymi, musi zawierać dodatkowe ograniczenia opisujące oferty substytucyjne. W przypadku CLP opisowi ograniczeń substytucyjnych podlegają wyłącznie oferty, gdyż substytucyjność *towarów* jest wyrażana za pomocą

ofert. Postać matematyczna ofert substytucyjnych wygląda następująco:

$$x_{j1} + x_{j2} + ... + x_{jS} = 1 \tag{5.3}$$

gdzie:
S - jest liczbą wszystkich ofert substytucyjnych, złożonych przez oferenta j-tego.

5.3.3 Ograniczenia komplementarne

Zgłaszane na aukcjach oferty mogą charakteryzować się również komplementarnością. Oznacza to, że iloczyn wszystkich zmiennych decyzyjnych złożonych ofert komplementarnych musi być wartością binarną 0 lub 1. Koniunkcja zmiennych binarnych równa się 1 w przypadku przyjęcia wszystkich ofert komplementarnych za wygrywające, lub 0 w przeciwnym przypadku. Odrzucenie którejkolwiek oferty komplementarnej jest równoznaczne z odrzuceniem wszystkich ofert komplementarnych tego samego oferenta. W przypadku ofert komplementarnych przyjętych jako wygrywające postać matematyczna tego wyrażenia wygląda następująco:

$$x_{j1} * x_{j2} * ... * x_{jK} = 1 \tag{5.4}$$

gdzie:
K - jest liczbą wszystkich ofert komplementarnych, złożonych przez j-tego oferenta.

Oferty komplementarne łączą w sobie dodatkowe ograniczenie związane z niepodzielnością ofert. Mówi ono, że złożone oferty muszą zostać przyjęte lub odrzucone w całości, to znaczy nie dopuszcza sie ich podziału. W przypadku składania pojedynczych ofert ograniczenie to zostaje automatycznie spełnione, dzięki binarnej właściwości zmiennej decyzyjnej opisanej wcześniej. Natomiast w przypadku składania większej liczby ofert z zastrzeżeniem ich komplementarności, fakt ten musi zostać dodatkowo ujęty w modelu reprezentującym problem.

5.4 Modelowanie i parametryzacja ograniczeń

Modelowanie i parametryzacja ograniczeń jest kluczowym elementem rozwiązywania problemów decyzyjnych za pomocą techniki programowania CLP. To właśnie od poprawności zamodelowanych ograniczeń zależy efektywność całego programu, rozwiązującego problem decyzyjny oraz poprawność otrzymanych wyników. Istnieje wiele różnych predykatów i metod umożliwiających modelowanie i reprezentację ograniczeń w technice CLP [Bar01]. Jednakże standardowe predykaty wbudowane w oprogramowanie CHIP v5.8, ze względu na swoje ograniczone właściwości, nie mogą być użyte do modelowania ograniczeń występujących w problemie WDP. Dlatego wszystkie metody oraz specjalistyczne predykaty wykorzystane do budowy ograniczeń problemu WDP zostały zaprojektowane przez autora pracy.

Modelowanie ograniczeń problemu WDP za pomocą CLP może zostać zrealizowane na dwa różne sposoby [Tat07]:

1. Pierwszy z nich to statyczne wpisywanie ograniczeń do modelu (matematycznych i logicznych zależności pomiędzy zmiennymi), co w dalszej części pracy będzie nazywane jako *statyczne modelowanie ograniczeń - SMO*. Modele CLP dla problemu WDP zbudowane w oparciu o statyczne modelowanie ograniczeń będą nazywane *SMO-WDP*.

2. Drugim sposobem modelowania ograniczeń jest sposób dynamiczny, na podstawie własnych predykatów tworzących w sposób automatyczny logiczne zależności zmiennych, które są podstawą tych ograniczeń. W dalszej części pracy ten sposób budowy ograniczeń będzie określany jako *dynamiczne modelowanie ograniczeń - DMO*, a modele zbudowane w oparciu o te ograniczenia będą opisywane jako *DMO-WDP*.

5.4.1 Statyczne modelowanie ograniczeń

Statyczne modelowanie ograniczeń ogólnych

Wszystkie opisane wcześniej ograniczenia występujące w problemie WDP muszą zostać bezpośrednio zaimplementowane w model CLP. Implementacja ta odbywać się może na dwa różne sposoby. Pierwszym z nich jest deklaracja ograniczeń w sposób statyczny. Oznacza to, że wszystkie ograniczenia występujące

w modelu są opisywanie za pomocą logicznych relacji zachodzących pomiędzy zmiennymi oraz domenami, a następnie relacje te są zapisywane w model problemu.
W przypadku aukcji jednostkowych wielu towarów modelowanie ograniczeń ogólnych odbywa się za pomocą form liniowych, reprezentujących poszczególne towary. Forma liniowa każdego ograniczenia jest tworzona na podstawie sumy binarnych zmiennych decyzyjnych, reprezentujących poszczególne oferty zawierające ten sam towar. Każdy wiersz tych ograniczeń przyrównywany jest do jedności, gdyż liczba dostępnych jednostek równa jest jeden. Kod źródłowy fragmentu modelu CLP realizujący statyczny opis ograniczeń problemu WDP dla danych z przykładu 2.1 wygląda następująco:

```
X1+X2 #<=1,
X3+X4 #<=1,
X1+X3 #<=1,
```

W przypadku aukcji wielu jednostek wielu towarów do statycznego modelowania ograniczeń niezbędna jest implementacja dodatkowych zmiennych, reprezentujących fizyczne jednostki danego towaru wystawionego na akcji. Forma liniowa takiego ograniczenia reprezentowana jest przez sumę iloczynów zmiennych decyzyjnych i jednostek znajdujących się w ofercie, którą dane zmienne reprezentują, i przyrównywana jest do całkowitej liczy jednostek wystawionych na aukcji. Fragment kodu źródłowego modelu CLP realizujący te ograniczenia dla danych z przykładu 2.3 wygląda nstępująco:

```
X1*2+X2*4 #<=6,
X3*2+X4*3 #<=4,
X1*3+X3*4 #<=5,
```

Statyczne modelowanie ograniczeń substytucyjnych

Ograniczenia substytucyjne, w przypadku statycznego ich modelowania, mogą wyglądać podobnie jak ograniczenia ofert wykluczających się. To znaczy ich zapis jest formą liniową, która to forma w przypadku ograniczeń substytucyjnych zawiera zmienne decyzyjne, reprezentujące oferty złożone przez tego samego ofertenta. Suma tych zmiennych jest przyrównywana do wartości jedności zarówno w przypadku aukcji jednostkowej wielu towarów, jak i wielu jednostek wielu towarów. Załóżmy, że dla omawianego przykładu 2.1 oferent

składający oferty oznaczone jako $Oferta1$ i $Oferta3$ zakłada ich substytucyjność. W statyczny sposób zapisujemy to ograniczenie w modelu CLP za pomocą aktywnego ograniczenia mającego na celu niedopuszczenie do przyjęcia obu złożonych ofert. Dla tego przypadku dodatkowe statycznie ograniczenie będzie wyglądało następująco:

```
X1+X3 #<=1
```

W przypadku gdy złożone na aukcji oferty objęte są ograniczeniem substytucyjnym, a posiadają towary wykluczające się, nie trzeba tego faktu dodatkowo modelować, gdyż oferty te wykluczają się wzajemnie poprzez tworzone ograniczenie komplementarne. Należy pamiętać natomiast, że ograniczenia substytucyjne są ograniczeniami dodatkowymi, a więc w przypadku gdy oferty substytucyjne nie posiadają towarów wykluczających się, należy ograniczenia te dodatkowo w modelu umieścić.

Omawiana metoda statycznego modelowania ograniczeń, jak każde rozwiązanie, posiada zalety i niedogodności. Do podstawowych niedogodności można zaliczyć fakt, iż statyczne opisywanie i deklarowanie ograniczeń w modelu rozwiązującym problem WDP wiąże się z koniecznością deklaracji bardzo dużej liczby zmiennych oraz ich relacji w modelu. Może to bezpośrednio prowadzić do popełnienia błędu podczas zapisu tych relacji, gdyż wiąże się to rozpisaniem ograniczeń dla form liniowych wszystkich towarów i ofert. W ten sposób powstaje macierz, której wiersze odpowiadają towarom wystawionym na aukcji, a kolumny ofertom.

Do niedogodności tego rozwiązania można również zaliczyć konieczność wprowadzenia dodatkowych wartości, odpowiadających liczbie wystawianych na aukcji towarów w przypadku aukcji wielu jednostek wielu towarów. Dodatkowo odpowiednie wartości, reprezentujące liczby jednostek towarów znajdujących się w ofertach, w przypadku tego typu aukcji należy również wprowadzić przy każdej zmiennej decyzyjnej. Zmienne te mogą występować również w modelu dla aukcji jednostkowych w postaci jedności, ale z punktu widzenia modelu nie ma to żadnego znaczenia, więc można je pominąć podczas tworzenia modelu.

Zaletą statycznego modelowania ograniczeń jest fakt, iż model nie musi

posiadać deklaracji danych wejściowych, omówionych w 5.2. W przypadku aukcji kombinatorycznej jednostkowej wielu towarów statyczne modelowanie ograniczeń umożliwia deklarację danych wejściowych tylko za pomocą zmiennych decyzyjnych oraz funkcji celu. W przypadku aukcji wielu jednostek wielu towarów, deklaracja danych wejściowych odbywa się za pomocą zmiennych decyzyjnych oraz liczby jednostek dostępnego towaru, a także funkcji celu.

Statyczne modelowanie ograniczeń komplementarnych

Statyczne modelowanie ograniczeń komplementarnych za pomocą programowania w logice z ograniczeniami nie jest już tak proste jak w przypadku statycznego modelowania ograniczeń substytucyjnych. Ograniczenia komplementarne można zamodelować podobnie jak ograniczenia substytucyjne, lecz w takim przypadku otrzymane rozwiązanie może nie być optymalne. Podczas statycznego modelowania ograniczeń ogólnych lub substytucyjnych suma binarnych zmiennych decyzyjnych musi być mniejsza lub równa wartości 1, a więc 1 lub 0. W przypadku statycznego modelowania ograniczeń komplementarnych suma wartości zmiennych decyzyjnych ofert objętych tymi ograniczeniami musi być równa liczbie tych ofert lub wartości 0. Statyczny model ograniczenia komplementarnego, gdzie suma wartości zmiennych decyzyjnych równa się liczbie ofert można zamodelować w następujący sposób:

```
a) X2+X3 #=2
```

Po tak postawionym ograniczeniu w otrzymanym rozwiązaniu na pewno będą się znajdywały obie oferty reprezentowane przez zmienne decyzyjne $X2$ oraz $X3$, gdyż powyższe ograniczenie musi zostać spełnione. Otrzymane rozwiązanie, jak już wcześniej wspomniano, może jednak nie być optymalne, gdyż po tak postawionym pierwszym ograniczeniu nie ma możliwości zamodelowania drugiego przypadku, a więc odrzucenia obu ofert. W tym bowiem przypadku ograniczenie to musiałoby wyglądać następująco:

```
b) X2+X3 #=0
```

Jak wiadomo oba te ograniczenia wykluczają się, a więc w modelu CLP, reprezentującym problem WDP, może się znaleźć tylko jedno z nich, aby zostało ono spełnione. Ograniczenia komplementarne ofert należy zatem uzyskać na inne drodze. Możliwe są dwa sposoby:

- Pierwszy z nich to wyznaczyć rozwiązanie optymalne bez ograniczeń komplementarnych (przypadek b), a następnie wyznaczyć rozwiązanie optymalne z ograniczeniami komplementarnymi (przypadek a) i przez porównanie obu otrzymanych wyników wybrać to rozwiązanie, które generuje większy zysk. Oczywiście wiąże się to z wielokrotnym zwiększeniem czasu wyznaczania rozwiązań, gdyż problem musi być wielokrotnie rozwiązywany.

- Drugim, znacznie prostszym sposobem modelowania komplementarności ofert, może być połączenie wszystkich ofert złożonych na aukcji przez tego samego oferenta i zamodelowania ich w modelu CLP jako jednej wspólnej oferty. W ten sposób można uniknąć modelowania ograniczeń komplementarnych za pomocą dodatkowych zmiennych i predykatów. Ograniczenia te zostały i tak postawione, gdyż albo oferta wspólna zawierająca wszystkie oferty komplementarne została ujęta jako wygrywająca, albo jako odrzucona. Ten sposób modelowania komplementarności ma swoje plusy, a mianowicie można dodatkowo tworzyć ograniczenia substytucyjne dla ofert komplementarnych, a także rozmiar problemu się zmniejsza poprzez zmniejszenie liczby ofert.

Reasumując, statyczne modelowanie ograniczeń komplementarnych za pomocą CLP najłatwiej osiągnąć poprzez połączenie wszystkich ofert komplementarnych złożonych przez tego samego oferenta. Ponieważ w przypadku statycznego modelowania ograniczeń nie deklarujemy ofert w postaci danych wejściowych lecz w postaci ograniczeń, dlatego szczegóły dotyczące łączenia ofert i ich deklaracja zostały opisane w dalszej części pracy.

5.4.2 Dynamiczne modelowanie ograniczeń

Dynamiczne modelowanie ograniczeń ogólnych

Dynamiczne modelowanie ograniczeń pozwala ominąć większość niedogodności związanych z ich statycznym modelowaniem. Dynamiczne modelowanie ograniczeń w modelu CLP dla problemu WDP jest możliwe dzięki zastosowaniu zaproponowanego przez autora predykatu rekurencyjnego $tworz\_ograniczenie\_ogolne\_sub(\_, \_, \_)$. Predykat ten podczas działania programu buduje ponownie, za pomocą kolejnego zaproponowanego przez autora predykatu $tworz\_liniowe(\_, \_, \_)$, formy liniowe poszczególnych

zmiennych, reprezentujące liczby jednostek danego towaru, a następnie na podstawie tych form, buduje odpowiednie ograniczenia. Wszystko to dzieje się w pełni dynamicznie podczas ukonkretniania przez program zmiennych w trakcie przeglądu drzewa decyzyjnego.

Niewątpliwym atutem dynamicznego modelowania wszystkich występujących w problemie WDP ograniczeń jest uniwersalność budowy i zastosowania predykatów *tworz_ograniczenie_ogolne_sub*(_, _, _) i *tworz_liniowe*(_, _, _). Pozwala to na budowę, w tym samym modelu, ograniczeń dla obu typów omawianych aukcji kombinatorycznych zarówno jednostkowych wielu towarów jak i wielu jednostek wielu towarów. Dzieje się tak, ponieważ predykat buduje dynamicznie ograniczenia na podstawie danych wejściowych, a więc dynamiczne modelowanie ograniczeń ogólnych nie wymaga dodatkowych deklaracji danych.
Od typu danych wejściowych zależą w tym przypadku budowane ograniczenia, a więc i rodzaj aukcji, której dotyczą. Zatem jeden model CLP, służący do rozwiązywania problemu WDP, oparty o dynamiczne modelowanie ograniczeń, pozwala na wyznaczanie rozwiązań dla obu typów aukcji kombinatorycznych, bez konieczności ingerencji w kod źródłowy modelu.

Dynamiczne modelowanie ograniczeń pozwala na modelowanie bardzo dużych aukcji, gdzie liczba ofert może być liczona w tysiącach. Dodatkowo dynamiczny sposób budowy tych ograniczeń nie zmniejsza czytelności kodu źródłowego programu, a wręcz powoduje znaczne zmniejszenie jego treści wraz z oddzieleniem deklaracji nazw i domen zmiennych od predykatów opisujących problem. Kod źródłowy modelu CLP dla problemu WDP realizujący dynamiczne budowanie form liniowych dla poszczególnych ograniczeń, niezależnie od liczby ofert wygląda następująco:

```
tworz_liniowe([],[],0).

tworz_liniowe([A|Li],[X|Vars],A*X+Liniowe_wyrazenie) :-
     !,
     tworz_liniowe(Li,Vars,Liniowe_wyrazenie).
```

Pozostała część kodu źródłowego modelu, realizująca dynamiczne budowanie ograniczeń ogólnych dla dowolnej aukcji kombinatorycznej oraz dla dowolnej liczby jej uczestników, wygląda następująco:

```
tworz_ograniczenie_ogolne_sub([],Vars,[]).

tworz_ograniczenie_ogolne_sub([],_,_) :-
     sprintf(Msg,"Blad: rozna dlugosc list",[]),
     writeln(Msg),
     fail.

tworz_ograniczenie_ogolne_sub(_,_,[]) :-
     sprintf(Msg,"Blad: rozna dlugosc list",[]),
     writeln(Msg),
     fail.

tworz_ograniczenie_ogolne_sub([Li|List],Vars,[Ui|Units]) :-
     !,
     tworz_liniowe(Li,Vars,Liniowe_wyrazenie),
     Liniowe_wyrazenie #<= Ui,
     tworz_ograniczenie_ogolne_sub(List,Vars,Units).
```

Zaproponowany przez autora predykat, służący do dynamicznej budowy ograniczeń ogólnych został nazwany *tworz_ograniczenie_ogolne_sub*, gdyż pozwala on dodatkowo na budowę dynamicznych ograniczeń substytucyjnych.

Dynamiczne modelowanie ograniczeń substytucyjnych

Dynamiczne modelowanie ograniczeń substytucyjnych w modelu CLP dla problemu WDP może odbywać się za pomocą dwóch dodatkowych predykatów, przechowujących dane opisujące te ograniczenia, oraz predykatów dynamicznie budujących te ograniczenia. Programowanie w logice z ograniczeniami, pozwala w bardzo prosty sposób modelować te ograniczenia w odróżnieniu od imparatywnych technik programowania, gdzie modelowanie dodatkowych ograniczeń wiązałoby się z koniecznością tworzenia skomplikowanego algorytmu.

W przypadku CLP kod źródłowy własnych predykatów, służących do dynamicznego budowania ograniczeń substytucyjnych, nie różni się od predykatów służących do dynamicznego budowania ograniczeń ofert wzajemnie wykluczających się, a więc od predykatów budujących ograniczenia ogólne. Dynamiczne modelowanie ograniczeń ogólnych odbywa się poprzez opisaną wcześniej analizę danych wejściowych. Predykat budujący dynamiczne

ograniczenia substytucyjne również wymaga pewnych dodatkowych danych wejściowych, na podstawie których ograniczenia te będzie budował.

W przypadku programowania w logice z ograniczeniami dane można najprościej zadeklarować za pomocą dwóch predykatów, zawierających listy tych danych *specyfikacje_sub*([]) oraz *jednostki_sub*([]). Dane są określone przez predykaty, których argumenty opisują zależności substytucyjne występujące pomiędzy ofertami. Pierwszy z tych predykatów ma na celu opisanie, które oferty są poddawane ograniczeniu substytucyjności, natomiast drugi opisuje wartości tych ograniczeń.
Kod źródłowy deklaracji specyfikacji, opisującej oferty poddawane ograniczeniom substytucyjnym, w przypadku obu typów aukcji kombinatorycznych, a więc jednostkowych wielu towarów oraz wielu jednostek wielu towarów wygląda tak samo:

```
specyfikacje_sub([[1,1,0,0,0],
                  [0,0,0,0,0],
                  [0,0,1,1,1],
                  [0,0,0,0,0],
                  [0,0,0,0,0]
                 ]).
```

Interpretacja powyższego zapisu jest następująca: na aukcji złożono pięć ofert, każda z tych ofert reprezentowana jest przez oddzielną listę. Numer każdej kolejnej listy odpowiada numerowi złożonej przez oferenta oferty. Każda z list zawiera tyle elementów, ile zostało złożonych na aukcji ofert, a numer pozycji elementu w liście odpowiada numerowi złożonej oferty. Wartością 1 reprezentowane są w liście tylko te oferty, które zostają poddawane dodatkowemu ograniczeniu substytucyjnemu, natomiast wartością 0 są reprezentowane wszystkie pozostałe oferty. A zatem powyższy zapis informuje nas, że oferty o numerach 1 i 2 zostały zgłoszone przez tego samego oferenta i obie te oferty będą poddane dodatkowemu ograniczeniu substytucyjnemu. Powyższy zapis informuje nas również, że oferty o numerach 3, 4, 5 zostały zgłoszone przez innego oferenta, i one także będą poddane omawianemu ograniczeniu.

```
jednostki_sub([1,0,1,0,0]).
```

W predykacie *jednostki_sub*([]) możemy określić w jaki sposób ograniczamy oferty. Interpretacja tego zapisu jest następująca: pozycja elementu

w predykacie $jednostki\_sub([])$ odpowiada numerowi listy predykatu $specyfikacje\_sub([])$. W przypadku, gdy pierwszym elementem listy predykatu $jednostki\_sub([])$ jest wartość 1, wszystkie modelowane oferty znajdujące się w pierwszej liście predykatu $specyfikacje\_sub([])$ są ofertami substytucyjnymi.

Reasumując, dynamiczne modelowanie ograniczeń substytucyjnych to tak naprawdę nic więcej jak tylko dodatkowe dynamiczne modelowanie ograniczeń ofert wykluczających się, a więc ograniczeń ogólnych. Zaproponowane dynamiczne modlowanie ograniczeń substytucyjnych wymaga umieszczenia w modelu CLP tylko dwóch dodatkowych predykatów, zawierających specyfikację ofert substytucyjnych oraz informację, że dotyczą one ograniczeń substytucyjnych.

Dynamiczne modelowanie ograniczeń komplementarnych

Dynamiczne modelowanie ograniczeń komplementarnych wiąże się, podobnie jak w przypadku dynamicznego modelowania ograniczeń substytucyjnych, z koniecznością deklaracji dodatkowych danych wejściowych, opisujących te ograniczenia.
Najpierw należy zadeklarować w predykacie $specyfikacje\_komp([])$, które oferty są komplementarne, a następnie w predykacie $jednostki\_komp([])$ komplementarność tę zapisać. Zapis i znaczenie obu predykatów jest identyczne jak w przypadku deklaracji substytucyjności z jedną małą różnicą. Predykat $jednostki\_komp([])$ zawiera poszczególne sumy ofert składanych przez tego samego oferenta ograniczanych jako komplementarne, na przykład:

```
specyfikacje_komp([[1,1,0,0,0],
                   [0,0,0,0,0],
                   [0,0,1,1,1],
                   [0,0,0,0,0],
                   [0,0,0,0,0]
                  ]).

jednostki_komp([2,0,3,0,0]).
```

W kolejnym kroku, do dynamicznego budowania ograniczeń komplementarnych wykorzystać można niewiele zmodyfikowane predykaty, służące do dynamicznego budowania ograniczeń ogólnych i substytucyjnych:

```
tworz_ograniczenie_komplementarne([],Vars,[]).

tworz_ograniczenie_komplementarne([],_,_) :-
     sprintf(Msg,"Blad: rozna dlugosc list",[]),
     writeln(Msg),
     fail.

tworz_ograniczenie_komplementarne(_,_,[]) :-
     sprintf(Msg,"Blad: rozna dlugosc list",[]),
     writeln(Msg),
     fail.

tworz_ograniczenie_komplementarne([Li|List],Vars,[Ui|Units]) :-
     !,
     tworz_liniowe(Li,Vars,Liniowe_wyrazenie),
     Liniowe_wyrazenie #= Ui,
     tworz_ograniczenie_komp(List,Vars,Units).
```

Jedyne różnice w stosunku do predykatów dynamicznie budujących ograniczenia ogólne i substytucyjne to ich nazwy, a także znak w bilansie jednostek danego towaru. Zmiana nazwy predykatów jest oczywista, gdyż nowo tworzone predykaty nie mogą posiadać nazw już wykorzystywanych przez inne predykaty. Znakiem służącym natomiast do bilansowania liczby jednostek towaru w przypadku ograniczeń ofert wzajemnie wykluczających się, a więc ograniczeń ogólnych, jest <=, gdyż suma wszystkich jednostek danego towaru znajdującego się w ofertach przyjętych za wygrywające musi się równać lub być mniejsza od liczby jednostek wystawionych na aukcji. W przypadku ograniczeń substytucyjnych znakiem tym ponownie jest <=, gdyż suma jednostek danego towaru w zgłoszonych ofertach substytucyjnych również musi równać się lub być mniejsza od 1. Dlatego ograniczenia substytucyjne można modelować za pomocą tych samych predykatów, co ograniczenia ogólne. W przypadku ograniczeń komplementarnych znak ten zostaje zamieniony na =.

Napotkane problemy z modelowaniem komplementarności ofert, opisane podczas statycznego modelowania ograniczeń komplementarnych, występują również w przypadku dynamicznego ich modelowania. Jeżeli zadeklarujemy komplementarność ofert, to znajdą się one w otrzymanym rozwiązaniu, gdyż towrzone ograniczenie mówi, że forma liniowa zawierająca zmienne decyzyjne o wartościach 1 lub 0 musi się równać sumie ofert komplementarnych.

Rozwiazaniem tego problemu może być połączenie w jedna wspólną ofertę, wszystkich ofert komplementarnych złożonych przez tego samego oferenta. Przykładem deklaracji takiego złączenia ofert może być następujący zapis:

- deklaracja ofert złożonych przez tego samego oferenta bez zgłaszania ich komplementarności:

```
specyfikacje([[1,0],
              [0,0],
              [1,0],
              [0,1],
              [0,1]
             ]).

ceny([100,150]).
```

- deklaracja ofert złożonych przez tego samego oferenta z uwzględnieniem ich komplementarności:

```
specyfikacje([[1],
              [0],
              [1],
              [1],
              [1]
             ]).

ceny([250]).
```

Podczas łączenia ofert komplementarnych należy pamiętać, że łączone oferty mogą się wykluczać, z jednym wyjątkiem. Dotyczy on aukcji wielu jednostek wielu towarów i przypadku, gdy suma jednostek danego towaru, umieszczonego we wszystkich ofertach złożonych jako komplementarne, przez tego samego oferenta, jest większa od liczby jednostek wystawionych na aukcji. W tym przypadku zgłoszenie tych ofert jako komplementarnych, a wiec i połączenie ich nie jest możliwe.

Z obu zaprezentowanych w pracy metod modelowania ograniczeń, dużo lepszym i bardziej uniwersalnym rozwiązaniem jest dynamiczne ich modelowanie. Rozwiązanie to pozwala uniknąć niedogodności oprogramowania CHIP

v5.8, związanego z maksymalną liczbą deklarowanych zmiennych, przez co możliwe jest modelowanie problemu WDP nawet dla bardzo dużej liczby towarów i ofert.
Dynamiczne modelowanie ograniczeń wpływa ponadto znacznie na czytelność modelu, gdyż sam model nie zawiera w kodzie źródłowym ręcznie wprowadzanych, logicznych relacji zawartych pomiędzy zmiennymi, a jedynie predykaty opisujące ich dynamiczną budowę.
Niewątpliwym atutem dynamicznego modelowania ograniczeń jest również fakt, że za pomocą tych samych predykatów można budować ograniczenia dla obu typów omawianych aukcji kombinatorycznych - jednostkowych i wielu jednostek wielu towarów, bez konieczności ingerencji w budowę predkatów i ich kod źródłowy.
Ostatnim i bardzo ważnym atutem dynamicznego modelowania ograniczeń jest fakt, że za pomocą tego samego predykatu można budować ograniczenia ogólne i substytucyjne, a predykaty budujące ograniczenia komplementarne są bardzo podobne w swojej budowie do predykatów budujących ograniczenia ogólne i substytucyjne. Dodatkowo wszystkie predykaty budujące ograniczenia korzystają z tego samego predykatu, służącego do tworzenia form liniowych dla tych ograniczeń.

5.5 Modelowanie funkcji celu

Oba przypadki statycznego i dynamicznego modelowania ograniczeń dla problemu wyznaczenia zwycięzcy wymagają jeszcze zamodelowania funkcji celu. Jej wartość wyznaczana jest na podstawie sumy wartości cen poszczególnych ofert pomnożonych przez binarną zmienną decyzyjną danej oferty. Dla przypadku statycznego modelowania ograniczeń funkcja celu może wyglądać następująco:

```
Zysk #= X1*100+X2*150+X3*100+X4*100,
```

W tym przypadku wartość zmiennej *Zysk* jest wpisywana statycznie do modelu CLP, w zależności od deklarowanych danych wejściowych. W przypadku dużej liczby ofert złożonych na aukcji, zapisanie takiej formy liniowej może okazać się trudne do realizacji. Dlatego lepszym rozwiązaniem jest dynamiczne budowanie funkcji celu.
W przypadku dynamicznej budowy ograniczeń, tworzona dynamicznie forma

liniowa jest podstawowym składnikiem funkcji celu. Zatem nie trzeba dodatkowych operacji, aby ta funkcja powstała dynamicznie. Reasumując, w tym wypadku funkcja celu jest stała, niezależna „bezpośrednio" od deklarowanych danych wejściowych, a jedynie od dynamicznie tworzonej formy liniowej tych ograniczeń:

```
Liniowe_wyrazenie #= Zysk,
```

5.6 Tworzenie modelu CLP dla problemu WDP

Każdy model CLP, umożliwiający rozwiązywanie problemu wyznaczenia zwycięzcy aukcji kombinatorycznej, możemy podzielić na dwie części. Pierwsza z nich to część deklaracyjna, zawierająca parametry programu, deklarację zmiennych oraz ich domen, a także reprezentację danych wejściowych. Druga część modelu, to część zawierająca szczegółowy opis problemu, na podstawie którego wyznaczane jest rozwiązanie problemu. Ponieważ parametryzacja zmiennych oraz ich domen jest tworzona na podstawie danych wejściowych, dlatego pierwszą część modelu możemy nazwać częścią zmienną, uzależnioną od tych danych. Druga, stała część modelu zawiera jednakowy i stały opis problemu WDP, dla aukcji jednostkowych i aukcji wielu jednostek wielu towarów (dotyczy to przypadku dynamicznej budowy ograniczeń).

W związku z podziałem na zmienną i stałą cześć modelu, można z powodzeniem uprościć etap tworzenia kompletnego modelu problemu. W tym celu można zastosować zewnętrzny program, który na podstawie danych wejściowych stworzy pierwszą, zmienną część modelu, a następnie dołoży jego stały fragment. Tak powstały kod źródłowy wystarczy zapisać w pliku tekstowym, który bezpośrednio jest wczytywany do kompilatora programu CLP. Ten sposób tworzenia modeli CLP dla problemu WDP umożliwia bardzo szybkie generowanie modeli dla różnych przypadków aukcji, a także zmniejsza ryzyko popełnienia błędu w danych, podczas zapisu modelu.
Do napisania programu zewnętrznego tworzącego model CLP najłatwiej jest wykorzystać programowanie strukturalne. Prawie wszystkie środowiska programistyczne, stosujące programowanie strukturalne, posiadają zaawansowane komponenty, umożliwiające łatwą transformację przetwarzanych danych, a także późniejszy odpowiednio sformatowany ich zapis w plikach tekstowych.

Oprogramowanie CHIP v5.8 umożliwia częściowe sterowanie procesem wyznaczania rozwiazań poprzez zmianę strategii numeracyjnych oraz poprzez zmianę wartości górnego ograniczenia. Oba te elementy sterujące mogą być parametryzowane podczas budowy modelu w programie zewnętrznym. Program zewnętrzny może również wstawić do modelu parametr odpowiadający za wyświetlanie wyników. Brak parametru *?- wflags(128).* w modelu skutkuje tym, że są wyświetlane wszystkie wyniki pośrednie, a więc rozwiązania dopuszczalne. Po wstawieniu do modelu powyższego parametru, wyświetlony zostanie tylko ostateczny, optymalny wynik końcowy.

5.6.1 Zmienna część modelu CLP

Jak już wcześniej wspomniano, zmienna część modelu CLP, umożliwiającego rozwiąznie problemu WDP, zależy od danych wejściowych. Budowa tej części modelu odbywa się na podstawie danych wejściowych. Ze względu na fakt, że każdy przypadek aukcji kombinatoryczej różni się danymi wejściowymi od innych przypadków, ten fragment modelu CLP musi zostać zbudowany za każdym razem od nowa. Oczywiście od programisty zależy, czy wybierze modelowanie statyczne, posiadające pewne niedogodności, czy też modelowanie dynamiczne. W przypadku wyboru modelowania statycznego w tym fragmencie, oprócz deklaracji zmiennych zawierających dane wejściowe oraz deklaracji wszystkich ograniczeń występujących w regułach aukcji, musi się znaleźć jeszcze funkcja celu, która w modelu statycznym również zależna jest od danych wejściowych. W przypadku modelowania dynamicznego w tej części modelu znajduje się tylko deklaracja zmiennych, zawierająca reprezentacje danych wejściowych, funkcja celu jest bowiem modelowana dynamicznie.

5.6.2 Stała część modelu CLP

Stała część modelu, opisująca problem WDP, zawiera zasadniczy mechanizm, umożliwiający rozwiązywanie omawianego problemu. Mechanizm ten łączy wcześniej zadeklarowane zmienne z predykatami wykonującymi przegląd przestrzeni decyzyjnej danej aukcji. Ta część modelu jest stała i niezmienna dla wszystkich przypadków aukcji kombinatorycznych, dlatego może być doklejana do zmiennej części modelu. W ten sposób może powstać kompletny model CLP dla problemu WDP.

5.7 Modele CLP dla problemu WDP

W pracy przedstawiono dwa modele umożliwiające rozwiązywanie problemu wyznaczenia zwycięzcy aukcji kombinatorycznej. Pierwszym z nich jest model SMO-WDP, bazujący na statycznym modelowaniu ograniczeń. Drugim modelem jest model DMO-WDP, bazujący na dynamicznie modelowanych ograniczeniach.

5.7.1 Model ze statycznymi ograniczeniami SMO-WDP

Kompletny kod źródłowy modelu SMO-WDP, napisany zgodnie z konwencją oprogramowania CHIP v5.8, umożliwiający rozwiązywanie problemu WDP w oparciu o statyczne modelowanie ograniczeń, dla przykładu zaprezentowanego w 2.3, a więc dla aukcji wielu jednostek wielu towarów wygląda następująco:

```
top:-
    Decyzje=[X1,X2,X3,X4],
    Decyzje::0..1,
    X::0..1000,
    Zysk::0..1000,

    X1*2+X2*4  #<=6,
    X3*2+X4*3  #<=4,
    X1*3+X3*4  #<=5,

    Zysk #= X1*100+X2*150+X3*100+X4*100,

    Temp + Zysk #= 1000,
    min_max(labeling(Decyzje),Temp),
    writeln(Decyzje),
    write('Zysk maksymalny: '),
    write(Zysk).

labeling([H|T]):-
    indomain(H),
    labeling(T).

labeling([]).
```

Różnice w kodzie źródłowym modelu, które należałoby wprowadzić podczas dostosowywania go do danych zawartych w przykładzie 2.1, to usunięcie liczb jednostek w ograniczeniach oraz zmiana liczby jednostek towarów.

5.7.2 Model z dynamicznymi ograniczeniami DMO-WDP

W przypadku dynamicznego modelowania ograniczeń, kompletny kod źródłowy modelu DMO-WDP, umożliwiający rozwiązanie problemu WDP dla przykładu zaprezentowanego w 2.3, a więc dla aukcji wielu jednostek wielu towarów, bez modelowania ograniczeń komplementarnych i substytucyjnych wygląda następująco:

```
ceny([100,150,100,100]).

specyfikacje([
     [2,4,0,0],
     [0,0,2,3],
     [3,0,4,0],
     [0,0,0,2],
     [0,0,0,1]
     ]).

jednostki([6,4,5,2,3]).

tworz_liniowe([],[],0).

tworz_liniowe([A|Li],[X|Vars],A*X+Liniowe_wyrazenie) :-
     !,
     tworz_liniowe(Li,Vars,Liniowe_wyrazenie).

tworz_ograniczenie([],Vars,[]).

tworz_ograniczenie([],_,_) :-
     sprintf(Msg,"Blad: rozna dlugosc list",[]),
     writeln(Msg),
     fail.

tworz_ograniczenie(_,_,[]) :-
     sprintf(Msg,"Blad: rozna dlugosc list",[]),
     writeln(Msg),
     fail.
```

```
tworz_ograniczenie([Li|List],Vars,[Ui|Jednostki]) :-
     !,
     tworz_liniowe(Li,Vars,Liniowe_wyrazenie),
     Liniowe_wyrazenie #<= Ui,
     tworz_ograniczenie(List,Vars,Jednostki).

top :-
     length(Decyzje,10),
     utime(_),
     Decyzje :: 0..1,
     Temp :: 0..100000,
     Zysk :: 0..100000,
     jednostki(Jednostki),
     ceny(Ceny),
     specyfikacje(Dane),
     tworz_ograniczenie(Dane,Decyzje,Jednostki),
     tworz_liniowe(Ceny,Decyzje,Liniowe_wyrazenie),
     Liniowe_wyrazenie #= Zysk,
     Temp + Zysk  #= 100000,
     min_max(labeling(Decyzje),Temp),
     writeln(Decyzje),
     write('Maximum:: '),writeln(Goal),
     writeln(decision-Decyzje).

labeling([H|T]):-
     indomain(H),
     labeling(T).

labeling([]).
```

Kod źródłowy modelu CLP dla przykładu 2.1, a więc przypadku aukcji jednostkowej wielu towarów, będzie się różnił od powyższego kodu jedynie deklaracją danych wejściowych, a więc predykatów `specyfikacje([])` i `jednostki([])`. Pozostała część modelu pozostanie taka sama dla obu typów aukcji.

Kompletne kody źródłowe opisanych w tej części pracy modeli SMO-WDP i DMO-WDP, z najbardziej efektywnymi ich konfiguracjami, opisanymi w punkcie 6.4, pozwalającymi na implementację addytywnych

ograniczeń substytucyjnych i/lub komplementarnych, zostały umieszczone w kończącym pracę dodatku C.

Rozdział 6

Analiza i testy modeli CLP

Rozdział zawiera opis i charakterystykę danych testowych, na podstawie których testowane były modele SMO-WDP i DMO-WDP, a także metody parametryzacji tych modeli w celu osiągnięcia większej ich efektywności. Szczegółowo przedstawiono sposoby przeprowadzania testów i badań modeli oraz zaprezentowano otrzymane wyniki.

6.1 Środowisko testowe

W celu określenia efektywności i charakterystyk czasowych prezentowanych w pracy modeli CLP dla problemu wyznaczenia zwycięzcy aukcji kombinatorycznych SMO-WDP i DMO-WDP, niezbędne było przeprowadzenie szeregu testów i analiz. Do tego celu został wykorzystany komputer firmy HP Compaq - model dx2200 Microtower. Komputer ten wyposażony został w procesor Intel Pentium 4 HT 3.06 GHz z 1024 MB zainstalowanej pamięci RAM. Jednakże, jak się okazało podczas testów, kompilator CHIP v5.8 nie wykorzystywał całej mocy obliczeniowej procesora, a jedynie około 50% jego możliwości. Efekt ten związany jest z faktem, że kompilator ten nie obsługuje wielowątkowości procesorów. Wielkość zainstalowanej pamięci RAM nie ma większego wpływu na czas wyznaczania rozwiązań, gdyż kompilator CHIP korzysta z pamięci poprzez tak zwany stos lokalny i globalny. Obie wartości stosu są deklarowane i rezerwowane podczas uruchomiania kompilatora. I tak w przypadku testowanych w dalszej części pracy modeli zostały one ustawione na 20 MB dla stosu lokalnego i 20 MB dla stosu globalnego.

6.2 Dane testowe

Przedstawione w pracy modele CLP poddane zostały testom, opartym na pseudoaukcyjnych danych, generowanych przez program *CATS* (ang. Combinatorial Auction Test Suite)[Gal]. Program ten powstał na Uniwersytecie Stanford. Umożliwia on generowanie danych testowych, które mogą służyć do testowania i porównywania efektywności znanych i ciągle powstających nowych programów i algorytmów, służących do rozwiązywania problemu WDP [LBPS00]. Przykład wynikowego pliku tekstowego wygenerowanego przez program *CATS*, zawierającego pseudolosowe dane testowe wygląda następująco:

```
%% File generated by CATS v.2.1Wed Nov  7 15:16:26 2007
%% The CATS webpage is http://robotics.stanford.edu/CATS
%% PARAMETER SETTINGS:
% Goods: 10; Bids: 5;
% Distribution: L2; Runs: 1; Seed: 1194452074
% Legacy distribution parameters:
% Number of goods chosen according to Random distribution.
% Price chosen according to Linear Random distribution
% (low_linearly = 1, high_linearly = 1000).

goods 10
bids 5
dummy 0
0 3098694 1 4 6 7 #
1 6920661 0 1 3 5 6 #
2 1056031 3 7 8 #
3 3126460 9 5 8 4 7 #
4 1789612 4 6 #
```

Sens danych jest następujący: na przykład wiersz (2 1056031 3 7 8 #) oznacza, że oferta numer 2 o wartości 1056031 obejmuje zgłoszone towary o numerach 3, 7, 8.
Oprogramowanie CATS służy do modelowania realistycznych środowisk aukcyjnych. Program umożliwia generowanie benchmarkowych danych testowych dla różnych rozkładów ofert, charakteryzujących różne sposoby ich tworzenia. Benchmarki te parametryzowane są między innymi za pomocą słów kluczowych, odpowiadających różnym rozkładom ofert: *arbitrary, arbitrary-npv, arbitrary-upv, matching, paths, regions, regions-npv, regions-upv, scheduling, L1, L2, L3, L4, L5, L6, L7, L8*, szczegółowo opisanych w [LBS06].

Przed uruchomieniem generowania danych testowych w programie CATS, użytkownik musi ustalić kilka podstawowych parametrów. Do parametrów tych należy między innymi rodzaj rozkładu ofert, liczba wystawianych na aukcji towarów oraz liczba złożonych ofert ich kupna. Oprogramowanie CATS pozwala również na ustawienie kilku dodatkowych parametrów, należą do nich między innymi: parametr określający stałą liczbę towarów umieszczanych w ofertach, parametr charakterystyki wycen dla złożonych ofert.
Najprostszym przykładem uruchomienia programu wraz z niezbędnymi parametrami może być wydanie w katalogu programu polecenia:

```
cats.exe -d arbitrary -n 1 -goods 500 -bids 2000 -int_prices
```

Takie przykładowe wywołanie programu wygeneruje dane testowe zgodnie z rozkładem ofert *arbitrary* (-d arbitrary), utworzony zostanie jeden plik wynikowy (-n 1) zawierający 2000 ofert (-bids 2000), a oferty zawierać będą pseudolosowe liczby towarów z zakresu 1-500, zgodne z rozkładem *arbitrary* (-goods 500), cena każdej oferty będzie liczbą całkowitą (-int_prices).

6.3 Transformacja danych testowych

Dane testowe wygenerowane przez system CATS posiadają specyficzny format, przedstawiony wcześniej. Aby dane te mogły zostać użyte do testów konkretnych modeli CLP, jako dane wejściowe, muszą one zostać przetransformowane do postaci formatu, jaki przyjmuje testowany model. W tym celu niezbędne okazało się stworzenie podprogramu, umożliwiającego budowanie modelu CLP, zawierającego zarówno definicje danych wejściowych opartych o wygenerowane przez program *CATS* dane, jak i kompletny kod źródłowy stałej części modelu, opisanej w punkcie 5.6. Program ten, na podstawie wynikowego pliku tekstowego programu *CATS*, generuje kompletny kod źródłowy modelu wraz z danymi wejściowymi, przetransformowanymi do niezbędnego przez model CLP formatu. Format ten, został szczegółowo opisany w punkcie 5.2 niniejszej pracy. Następnie dodawana jest do pliku tekstowego danego modelu stała część kodu źródłowego, zawierająca wszystkie niezbędne predykaty opisujące problem. W ten sposób powstaje kompletny model będący plikiem tekstowym. Plik taki należy wczytać do kompilatora CHIP v5.8 i uruchomić.

6.4 Parametryzacja modeli CLP

Zaprezentowane w pracy modele SMO-WDP i DMO-WDP, w zależności od potrzeb, mogą ulegać wielu modyfikacjom. Modyfikacje te mają na celu zwiększenie ich efektywności, a więc ograniczenie czasu wyznaczania rozwiązania optymalnego. Można to uzyskać poprzez zmniejszenie drzewa decyzyjnego rozwiązywanego problemu. Do metod zmniejszania drzewa decyzyjnego w paradygmacie programowania w logice z ograniczeniami zaliczyć można dwa elementy: zmianę strategii numeracyjnej oraz zmianę metody ukonkretniania zmiennych. Przed rozpoczęciem szczegółowych testów omawianych modeli należy zatem ustalić możliwie najlepsze parametry pracy tych modeli.

W praktyce większą efektywność prezentowanych w pracy obu modeli CLP można usyskać między innymi stosując zamiast jednoargumentowego predykatu *labeling*(_) jego czteroargumentową wersję *labeling*(_, _, _, _). Różnica pomiędzy nimi polega na tym, że w przypadku czteroargumentowej wersji *labeling*u istnieje możliwość wprowadzenia zmiany strategii numeracyjnej, pozwalającej na rozpoczynanie i przeglądanie przestrzeni decyzyjnej problemu na różne sposoby. Szczegóły dotyczące wszystkich możliwych do zastosowania w predykacie *labeling* strategii numeracyjnych zostały opisane w punkcie 4.2.3.

Jak już wcześniej wspomniano, wybór najlepszej strategii numeracyjnej dla danego problemu decyzyjnego odbywa się na zasadzie *wybierz i testuj*. Dlatego, w celu określenia najlepszej strategii numeracyjnej dla omawianego problemu, przeprowadzono szereg testów. Testy te oparte zostały o dane testowe dla przykładowej jednostkowej aukcji kombinatorycznej z 20 wystawionymi na aukcji towarami i 200 zebranymi ofertami ich kupna. Otrzymane wyniki przedstawione zostały odpowiednio: w tabeli 6.1 dla modelu SMO-WDP oraz w tabeli 6.2 dla modelu DMO-WDP.

Jak widać najlepszą strategią numeracyjną, która najszybciej doprowadziła do optymalnego rozwiązania przykładowej aukcji, okazała się strategia *most_constrained*. W związku z tym, w dalszej części niniejszej pracy, wszystkie modele poddawane testom i analizom, będą posiadały zaimplementowaną tę właśnie, najbardziej efektywną dla problemu wyznaczenia zwycięzcy, strategię numeracyjną.

Tabela 6.1: Porównanie efektywności strategii numeracyjnych w modelach SMO-WDP

Strategie numeracyjne	Czas wyznaczania rozwiązania optymalnego (sek.)
smallest	0,536
largest	0,541
most_constrained	0,528
first_fail	14,313
max_regret	0,536

Tabela 6.2: Porównanie efektywności strategii numeracyjnych w modelach DMO-WDP

Strategie numeracyjne	Czas wyznaczania rozwiązania optymalnego (sek.)
smallest	0,547
largest	0,549
most_constrained	0,531
first_fail	14,343
max_regret	0,547

Obok ustalenia najlepszej strategii numeracyjnej, należy również dobrać najlepszą metodę ukonkretniania zmiennych. Domyślnie jednoargumentowy *labeling*(_) korzysta z jednoargumentowego predykatu *indomain*(_), mającego na celu ukonkretnianie zmiennych. Wprowadzenie czteroargumentowej wersji predykatu *labeling*(_, _, _, _) pozwala na zastosowanie dwuargumentowej wersji predykatu *indomain*(_, _). Za pomocą drugiego argumentu tego predykatu można wpływać na zmianę metody ukonkretniania zmiennych, co skutkuje zmianą dokonywanych dopasowań wartości do zmiennych. Wprowadzenie powyższych modyfikacji w kodzie źródłowym modelu CLP może wyglądać następująco:

```
min_max((
     labeling(Decyzje,0,most_constrained,fix),
     get_choice(C)),Temp).

get_choice(C):-
     getval(choice,N),
     setval(time,N),
     write(N),
     inc_choice.

inc_choice:-
     incval(choice,_),
     fail.
     inc_choice.

fix(Temp):-
     indomain(Temp,max).
```

W przypadku jednoargumentowej wersji predykatu *indomain*(_), jedynym argumentem tego predykatu jest domena zmiennych poddawana ukonkretnieniu. Zmienne, w tym przypadku, są ukonkretniane dopuszczalnymi wartościami od najniższej do najwyższej dla danej domeny. W przypadku dwuargumentowej wersji tego predykatu, pierwszym argumentem jest ponownie domena zmiennych poddawana ukonkretnieniu, natomiast drugi argument może przyjmować jedną z trzech wartości, reprezentującą różne metody ukonkretniania: *min*, *middle*, *max*.
Metoda *min* pozwala na ukonkretnianie zmiennych tak samo jak jednoargumentowy predykat *indomain(_)*, a więc zmiennne danej domeny są uzupełniane wartościami od najniższej, dopuszczalnej w danej domenie, do najwyższej. Metoda *middle* przydziela wartości zmiennym, na przemian powyżej i poniżej średniej dopuszczalnej wartości danej domeny. Natomiast metoda *max* przydziela wartości zmiennym począwszy od największej dopuszczalnej wartości danej domeny do najmniejszej.

W celu określenia, która metoda ukonkretniania zmiennych predykatu *indomain*(_, _) jest bardziej efektywna w modelach SMO-WDP i DMO-WDP, przeprowadzono testy porównawcze tych metod, podobnie jak w przypadku porównywania różnych strategii numeracyjnych. Testy te przeprowadzono w oparciu o dane testowe, wygenerowane w rozkładzie *arbitrary*

dla jednostkowej aukcji kombinatorycznej z 20 towarami wystawionymi do sprzedaży i 500 zebranymi na aukcji ofertami ich kupna. Wyniki z przeprowadzonych testów zostały zamieszczone w tabeli 6.3 dla modeli SMO-WDP oraz w tabeli 6.4 dla modeli DMO-WDP.

Tabela 6.3: Metody i czasy ukonkretniania zmiennych - modele SMO-WDP

Metoda ukonkretniania	Czas (sek.)	Liczba wyznaczanych rozwiązań dopuszczalnych
min	39,941	211
middle	39,289	211
max	3,691	12

Tabela 6.4: Metody i czasy ukonkretniania zmiennych - modele DMO-WDP

Metoda ukonkretniania	Czas (sek.)	Liczba wyznaczanych rozwiązań dopuszczalnych
min	40,270	211
middle	39,431	211
max	3,713	12

Podane w tabelach 6.3, 6.4 czasy są czasami wyznaczania rozwiązania optymalnego. Liczba wyznaczanych rozwiązań dopuszczalnych jest liczbą tych rozwiazań, które są wyznaczane przed uzyskaniem optymalnego rozwiązania. W rozpatrywanych trzech przypadkach otrzymane rozwiązania optymalne są takie same.

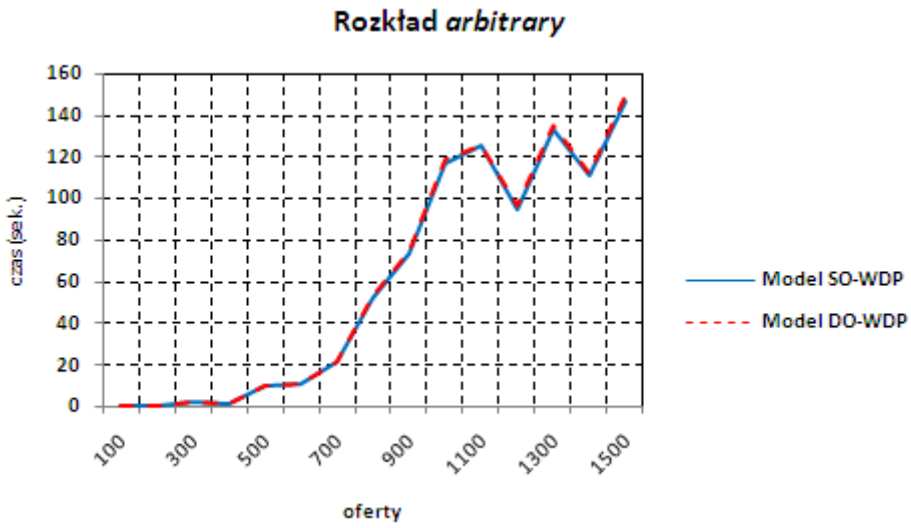
Zarówno jednoargumentowy predykat *indomain(_)*, ukrycie korzystający z metody *min*, jak i metody *min* i *middle* dwuargumentowego predykatu *indomain(_,_)*, powodują bardzo szczegółowe przeglądanie drzewa decyzyjnego, co skutkuje wyznaczaniem bardzo dużej liczby rozwiązań dopuszczalnych oraz wydłużeniem czasu wyznaczania rozwiązania optymalnego.

Reasumując, po przeprowadzeniu testów dla wszystkich metod ukonkretniania zmiennych, modele SMO-WDP i DMO-WDP z zaimplementowaną metodą *max* okazały się najbardziej efektywne. Zastosowanie tej właśnie metody do ukonkretniania zmiennych prowadzi do znacznego zmniejszania drzewa decyzyjnego podczas działania programu, co w dużym stopniu ogranicza liczbę wyznaczanych rozwiązań dopuszczalnych, a w rezultacie końcowym, prowadzi do zmniejszenia całkowitego czasu niezbędnego do wyznaczenia rozwiązania lub rozwiązań optymalnych. W związku z tym opisane w dalszej części pracy testy i analizy będą oparte o modele z zaimplementowanym dwuargumentowym predykatem $indomain(\_, \_)$ i metodą ukonkretniania *max*.

Co ciekawe, przeprowadzone testy pokazały, że metoda modelowania ograniczeń i funkcji celu nie ma wpływu na liczbę wyznaczanych rozwiązań (rozwiązań dopuszczalnych). Modele SMO-WDP i DMO-WDP z ustawionymi takimi samymi argumentami predykatów, wyznaczały dokładnie te same liczby rozwiązań dopuszczalnych, a otrzymane rozwiązania optymalne były identyczne. Podobnie czasy wyznaczania rozwiązań optymalnych okazały się dla obu modeli bardzo zbliżone. Jeżeli metoda modelowania ograniczeń i funkcji celu nie wpływa na liczbę wyznaczanych rozwiązań, a uzyskiwane czasy wyznaczania rozwiązań optymalnych przez oba modele okazały się bardzo zbliżone, to można postawić pytanie, czy metoda modelowania ograniczeń i funkcji celu wpływa na efektywność tych modeli.

6.5 Porównanie efektywności modeli SMO-WDP i DMO-WDP

W celu odpowiedzi na postawione wcześniej pytanie, dotyczące wpływu metody modelowania ograniczeń i funkcji celu na efektywność tych modeli, należy przeprowadzić szczegółowe testy porównawcze. Zarówno model SMO-WDP jak i model DMO-WDP zastosowano do tych samych danych testowych z zakresu 100-1500 ofert oraz ze stałą liczbą 10 towarów wystawionych na aukcji do sprzedaży. Otrzymane wyniki zostały przedstawione na rysunku 6.1.



Rysunek 6.1: Porównanie efektywności modeli SMO-WDP i DMO-WDP

Po przeprowadzeniu testów okazało się, że oba modele SMO-WDP i DMO-WDP charakteryzują się prawie identyczną efektywnością. W każdym z przeprowadzonych testów, o ułamki sekund lepszą efektywność charakteryzował się model SMO-WDP, jednakże różnice te były tak małe, że mówić możemy o identycznych efektywnościach obu modeli. W celu potwierdzenia tych wyników, przeprowadzono również losowe testy dla innych rozkładów, za każdym razem otrzymując bardzo zbliżone wyniki. Reasumując, można zatem powiedzieć, że wybór metody modelowania ograniczeń i funkcji celu, nie ma wpływu na efektywność tych modeli, lecz wpływa jedynie na wygodę programowania i przejrzystość modelu.

6.6 Badanie modeli DMO-WDP

W wyniku przeprowadzenia analizy porównawczej obu modeli okazało się, że posiadają one niemal identyczną efektywność. W związku z tym nie ma potrzeby przeprowadzania szczegółowych dalszych testów obu modeli, a wyłącznie jednego z nich.
Zatem do dalszych szczegółowych testów i analiz został wybrany model DMO-WDP. Wybór modelu DMO-WDP nie jest przypadkowy, gdyż modele oparte

w swojej budowie o dynamiczne tworzenie ograniczeń i funkcji celu charakteryzują się:

- prostszą budową w porównaniu z modelem SMO-WDP,
- odzieleniem danych wejściowych od predykatów rozwiązujących problem,
- możliwością rozwiązywania zarówno problemów aukcji kombinatorycznych jednostkowych wielu towarów, jak i wielu jednostek wielu towarów.

Opisane w dalszej części pracy testy i analizy podzielone zostały na dwie grupy: pierwsza z nich dotyczy aukcji jednostkowych wielu towarów, druga aukcji wielu jednostek wielu towarów. Przeprowadzenie szczegółowych badań i testów modeli ma na celu określenie ich efektywności w różnych środowiskach aukcyjnych. Różne środowiska aukcyjne charakteryzują się różnymi rozkładami ofert, a także różnymi liczbami towarów i ofert branych pod uwagę w testach.

6.6.1 Aukcje jednostkowe wielu towarów

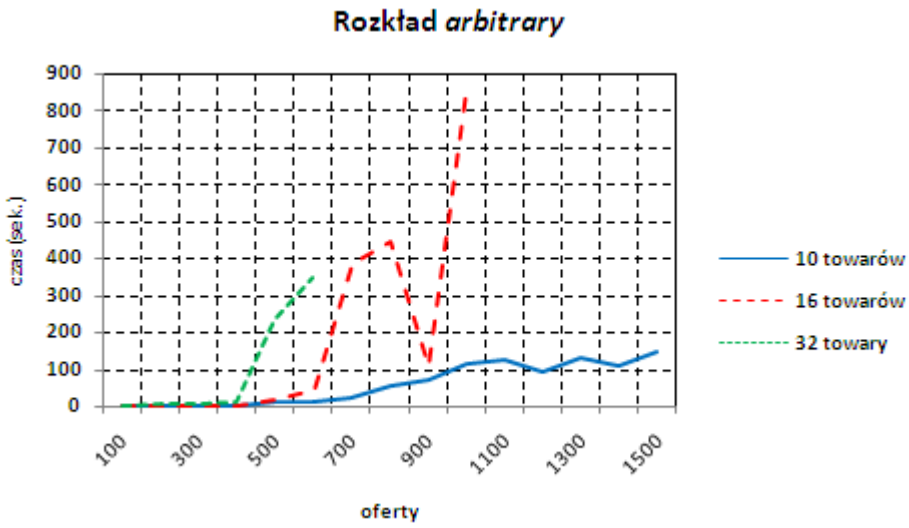
Badanie wpływu rozkładu ofert danych testowych na czas wyznaczania rozwiązania optymalnego

Pierwszy z rozpatrywanych przypadków dotyczy określenia wpływu zmiany rozkładu ofert danych testowych, wygenerowanych przez program *CATS*, na czas wyznaczania rozwiązania optymalnego testowanych modeli DMO-WDP. Do tego celu wygenerowane zostały zestawy danych testowch z następującymi rozkładami ofert *arbitrary*, *L2*, *L5*. W każdym z tych zestawów, znajdowało się odpowiednio 10 towarów, 16 towarów i 32 towary wystawione do sprzedaży na aukcji.

W przypadku danych o rozkładzie *arbitrary*, mała liczba towarów wystawionych do sprzedaży (10 towarów) i rosnąca liczba składanych ofert ich kupna powoduje niewielki wzrost czasu, jaki jest niezbędny do wyznaczenia rozwiązania optymalnego przez testowane modele. Wzrostowi liczby wystawionych na aukcji towarów do testowanych wartości 16 i 32, towarzyszy natomiast gwałtowny wzrost czasu wyznaczania rozwiązania optymalnego i to już przy około 400 złożonych ofertach. Wyniki przeprowadzonych testów modeli DMO-WDP, w oparciu o dane testowe wygenerowane w tym rozkładzie, przedstawione zostały w postaci wykresu na rysunku 6.2.

Na wykresie tym można dodatkowo zauważyć, że gwałtowny wzrost czasu, potrzebny do wyznaczenia rozwiązania optymalnego, nie jest stały, lecz podlega wahaniom. Wahania te są bardzo dobrze widoczne w przypadku 16 towarów i 900 ofert.

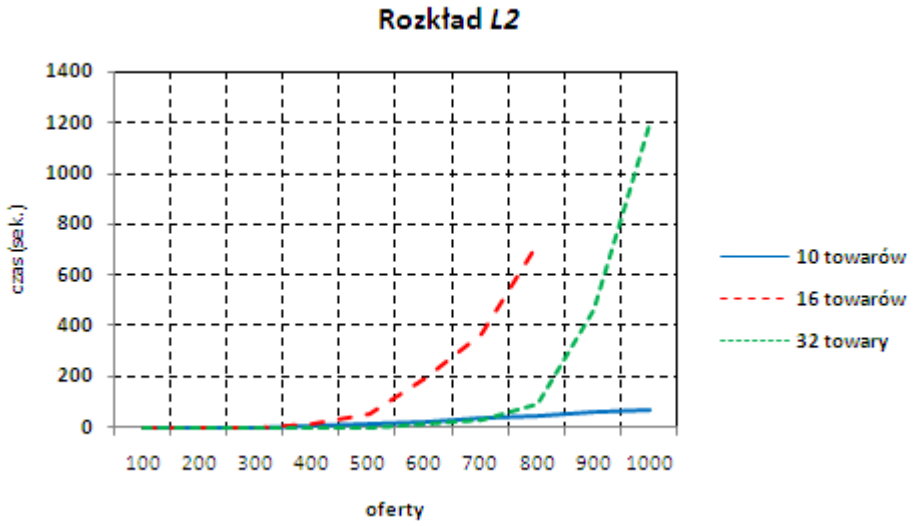
Uwaga: nagłe zakończenie linii na prezentowanych w pracy wykresach (przed osiągnięciem ich maksymalnej liczby ofert) oznacza, że koleja próba testowa dla danego rozkładu i zwiększonej liczby ofert, wymaga bardzo dużego wzrostu czasu, niezbędnego do wyznaczenia rozwiązania optymalnego. Dlatego, aby nie pogarszać czytelności prezentowanych w pracy wykresów, dalsze wartości nie są na nich prezentowane.



Rysunek 6.2: Czasy wyznaczania rozwiązań optymalnych modeli DMO-WDP dla danych testowych o rozkładzie *arbitrary*

W kolejnym etapie modele DMO-WDP poddane zostały testom w oparciu o dane testowe o rozkładzie *L2*. W tym przypadku okazało się, że niska liczba 10 towarów, przy zwiększającej się liczbie ofert, powoduje niemal stały (lekko rosnący) czas wyznaczania rozwiązania. W przypadku rozkładu *L2* dla 10 towarów nie było możliwe wyznaczenie rozwiązania dla liczby ofert większej niż 1000. Dlatego wykres znajdujący się na rysunku 6.3 prezentuje wyniki przeprowadzonych testów dla danych z zakresu 100 - 1000 ofert, a nie jak w przypadku

rozkładu *arbitrary* z zakresu 100 - 1500 ofert.



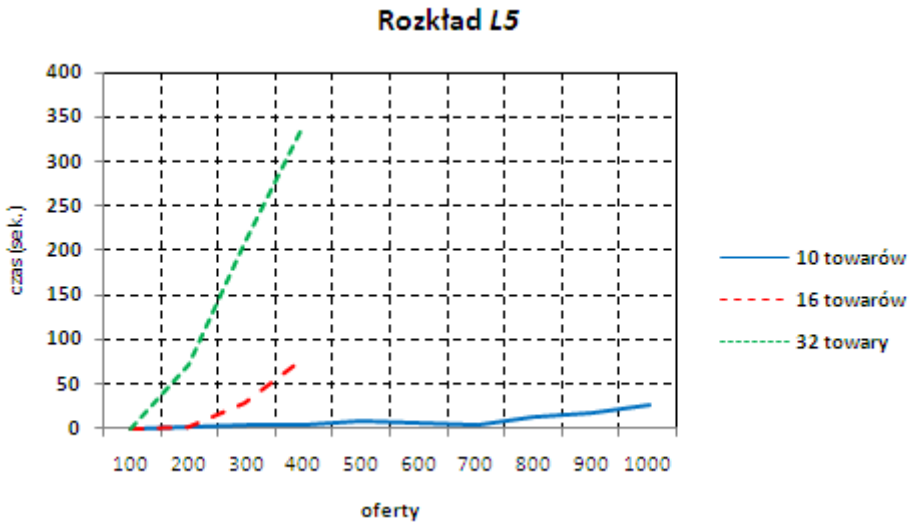
Rysunek 6.3: Czasy wyznaczania rozwiązań optymalnych modeli DMO-WDP dla danych testowych o rozkładzie *L2*

Zwiększenie liczby towarów do 16, a następnie do 32, z jednoczesnym wzrostem liczby ofert, podobnie jak w przypadku rozkładu *arbitrary*, powoduje gwałtowniejszy wzrost badanego czasu w stosunku do 10 towarów. Wzrost ten posiada jednak charakter bardziej uporządkowany bez widocznych w rozkładzie *arbitrary* wahań.

Ostatnim z przeprowadzonych testów modeli DMO-WDP, badających wpływ zmiany rozkładu ofert danych testowych na czas wyznaczania rozwiązania optymalnego, była zmiana tego rozkładu dla generowanych danych na *L5*. Podobnie jak w przypadku rozkładu *L2*, wygenerowane zostały dane z przedziału 100 - 1000 ofert. W wyniku przeprowadzenia testów powstało zestawienie w postaci wykresu, przedstawione na rysunku 6.4.

Otrzymane wyniki są niemal identyczne z wynikami dla rozkładu *L2*, z jedną małą różnicą. Przypadek wzrostu liczby towarów do 16 i 32 wraz z jednoczesnym wzrostem liczby ofert, generuje dużo większy wzrost czasu wyznaczania rozwiązania optymalnego. W przypadku rozkładu *L2* i 16 towarów, wzrost ten rozpoczyna się przy około 500 ofertach, natomiast

w przypadku rozkładu *L5* już przy około 300 ofertach. Jeszcze większa różnica jest w przypadku 32 towarów. Przy rozkładzie *L2* dynamiczny wzrost czasu można zauważyć przy 800 ofertach, natomiast przy rozkładzie *L5* praktycznie powyżej 100 ofert obserwuje się bardzo dynamiczny wzrost czasu.



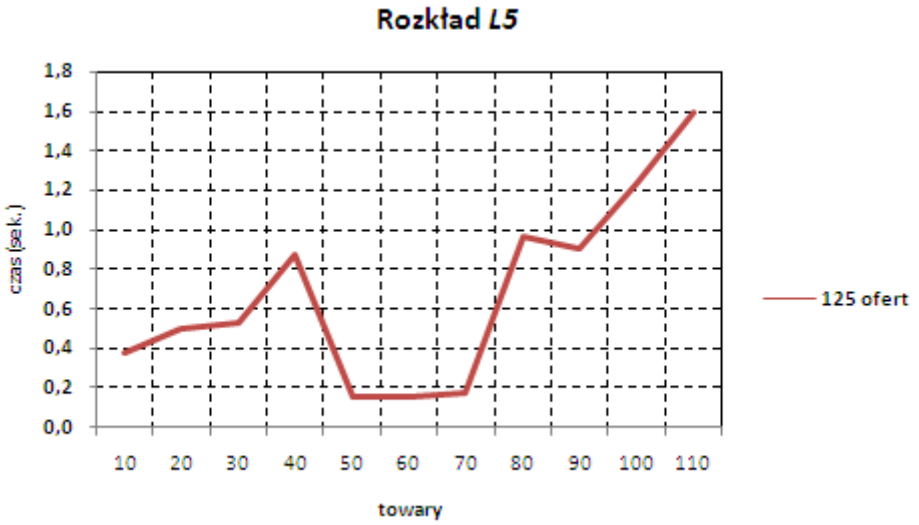
Rysunek 6.4: Czasy wyznaczania rozwiązań optymalnych modeli DMO-WDP dla danych testowych o rozkładzie *L5*

Przeprowadzone badanie wpływu zmian rozkładu ofert danych testowych na modele DMO-WDP jednoznacznie wskazują, że testowane modele wrażliwe są na zwiększanie liczby towarów, natomiast wykazują niewielką wrażliwość na zmiany rozkładu ofert. Bardzo dobre wyniki modele te uzyskują jednak w przypadku rozkładu *L5* i małej liczby towarów. W tym przypadku, uzyskują lepszą efektywność w porównaniu z wynikami zaprezentowanymi w cytowanej już pracy [GL01].

Badanie wpływu zmiany liczby towarów na efektywność modeli

Po przeprowadzeniu szeregu testów, mających na celu określenie wpływu zmian rozkładu ofert na efektywność modeli, okazało się, że modele te charakteryzują się małą efektywnością w przypadku małej liczby towarów i gwałtownie rosnącej liczby ofert. Kolejne badanie odpowiada przypadkowi odwrotnemu, dla stałej liczby ofert równej 125 i liczby towarów zmiennej

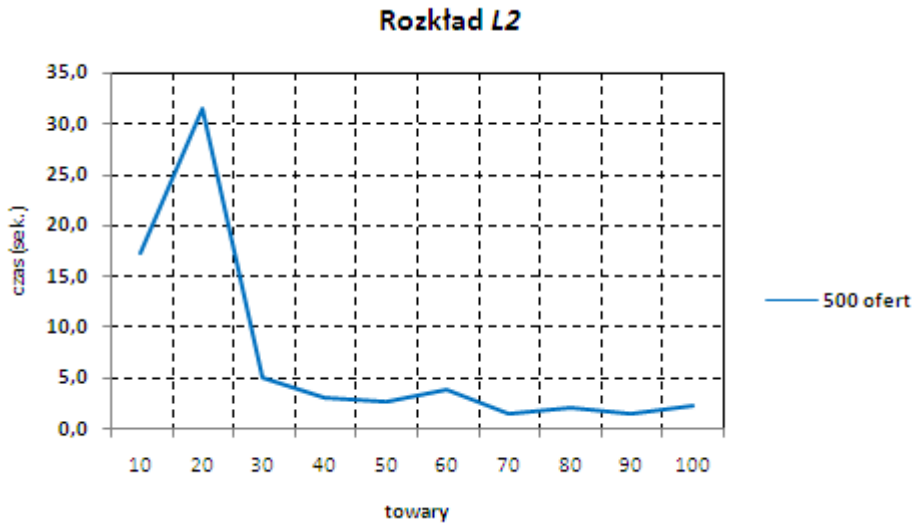
w przedziale 10-110. Wyniki z przeprowadzonych testów zostały przedstawione w postaci wykresu na rysunku:6.5



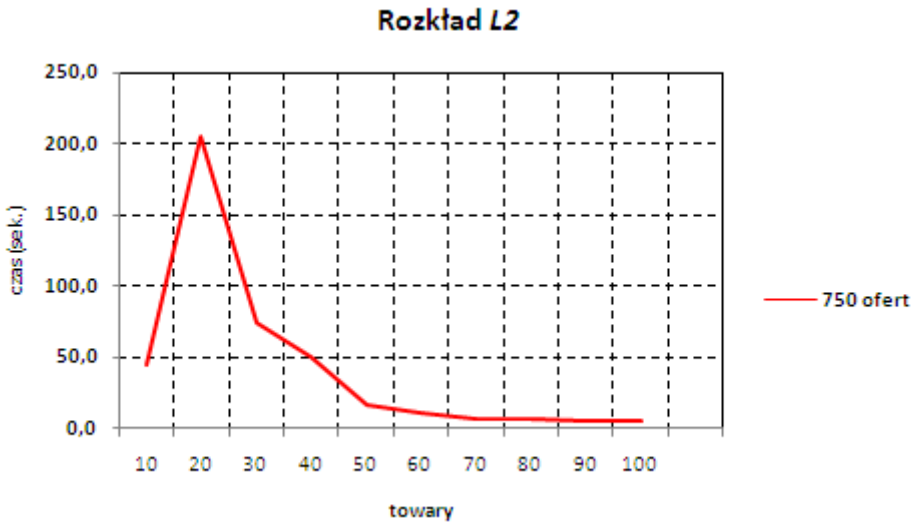
Rysunek 6.5: Czasy wyznaczania rozwiązań optymalnych modeli DMO-WDP dla danych testowych o rozkładzie *L5*

Przedstawiony na wykresie 6.5 wzrost liczby towarów przy stałej liczbie ofert spowodował niewielki wzrostu czasu wyznaczania rozwiązania optymalnego (od 0,5 sekundy do 1,8 sekundy). Wszystkie uzyskane w testach rozwiązania zostały wyznaczone w czasie poniżej 2 sekund, co jest wynikiem bardzo dobrym w porównaniu z wynikami opisanymi w pracy [GL01], gdzie wyniki podobnego testu oscylowały w okolicach od około 0,3 sekundy w przypadku 10 towarów do około 100 sekund w przypadku 110 towarów.

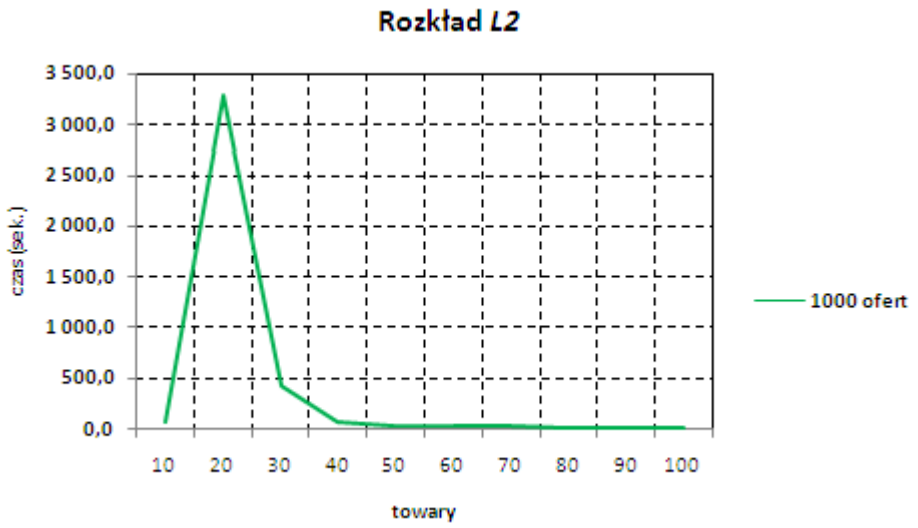
W tym miejscu można postawić zatem kolejne pytanie, a mianowicie jak stosunek liczby ofert do liczby towarów wpływa na czas wyznaczania rozwiązania lub rozwiązań optymalnych testowanych modeli. Aby udzielić odpowiedzi na powyższe pytanie przeprowadzone zostały kolejne testy. Tym razem wygenerowane zostały modele zawierające dane testowe z kolejno 500, 750, 1000, 1250 ofertami. Każdy z tych modeli zawierał kolejno 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 towarów wystawionych na aukcji, a rozkład ofert został ustawiony na *L2*.



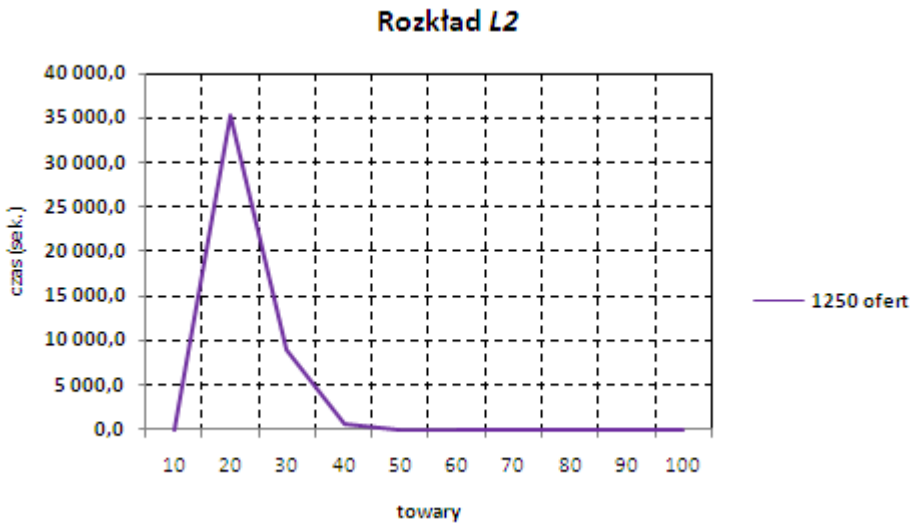
Rysunek 6.6: Czasy wyznaczania rozwiązań optymalnych modeli DMO-WDP dla danych testowych o rozkładzie *L2*



Rysunek 6.7: Czasy wyznaczania rozwiązań optymalnych modeli DMO-WDP dla danych testowych o rozkładzie *L2*



Rysunek 6.8: Czasy wyznaczania rozwiązań optymalnych modeli DMO-WDP dla danych testowych o rozkładzie *L2*



Rysunek 6.9: Czasy wyznaczania rozwiązań optymalnych modeli DMO-WDP dla danych testowych o rozkładzie *L2*

Wyniki z przeprowadzonych testów, w celu lepszej ich czytelności, zostały przedstawione w formie czterech oddzielnych wykresów na rysunkach 6.6, 6.7, 6.8 i 6.9. Na wszystkich prezentowanych wykresach można zauważyć pewne charakterystyczne cechy, które posiadają testowane modele, a mianowicie:

- wszystkie modele w przypadku stałej liczby ofert wyznaczają rozwiązania w bardzo zbliżonym czasie, z wyjątkiem modeli zawierających 20 i 30 towarów wystawionych do sprzedaży,
- wszystkie modele aukcji zawierające 20 towarów wykazują bardzo duże odchylenie od pozostałych osiągniętych czasów wyznaczania rozwiązań,
- osiągnięty najgorszy czas wyznaczania rozwiązań w przypadku 20 towarów maleje w momencie dodawania liczby towarów i stabilizuje się po osiągnięciu 40 towarów.
- wszystkie modele zawierające powyżej 20 towarów posiadają tendencję spadkową czasu wyznaczania rozwiązań,
- wszystkie modele zawierające poniżej 20 towarów posiadają tendencję wzrostową czasu wyznaczania rozwiązań,

Po przeprowadzeniu testów okazało się, że czas wyznaczenia rozwiązania lub rozwiązań optymalnych nie jest monotoniczną funkcją liczby towarów.
Teoretycznie trudniejsze przypadki aukcji, zawierające większą liczbę towarów są szybciej rozwiązywane przez prezentowane w pracy modele CLP niż przypadki łatwiejsze, z mniejszą liczbą towarów. Przyczyna tego faktu związana jest z różną efektywnością przeglądania drzewa decycyjnego danego problemu.

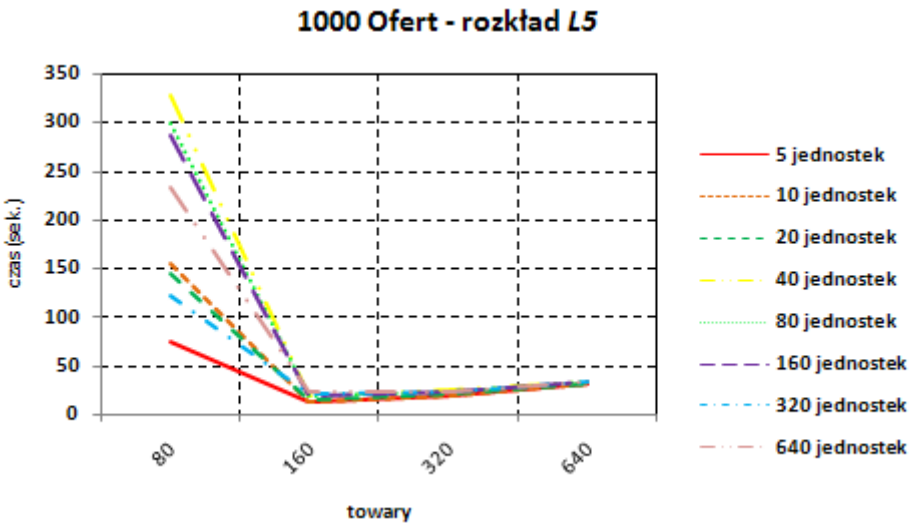
6.6.2 Aukcje wielu jednostek wielu towarów

Dla aukcji jednostkowych wielu towarów istnieje oprogramowanie *CATS*, umożliwiające generowanie danych testowych. Z oprogramowania tego korzystano w rozdziele 6.6.1.
W przypadku aukcji wielu jednostek wielu towarów takiego oprogramowania brak. Dlatego, aby było możliwe przetestowanie modeli DMO-WDP dla aukcji wielu jednostek wielu towarów, został napisany przez autora program, którego celem jest zmiana zbudowanych modeli SMO-WDP opartych o dane testowe

wygenerowane przez program CATS, na modele DMO-WDP dla aukcji wielu jednostek wielu towarów. Dokonywana przez program zmiana w modelach SMO-WDP polega tylko i wyłącznie na zmianie danych wejściowych. Zmiana ta realizowana jest przez wstawienie, w już istniejacym modelu dla aukcji jednostkowych, pseudolosowych liczb z zakresu od 1 do maksymalnej dostępnej liczby jednostek danego towaru w miejsca, gdzie w modelu znajdują się wartości pojedynczych jednostek towarów. Dodatkowo program tworzy predykat *jednostki()*, w którym to predykacie umieszcza fizycznie dostępne maksymalne liczby jednostek poszczególnych towarów. W ten sposób powstają modele zawierające dane testowe dla aukcji kombinatorycznej wielu jednostek wielu towarów. W dalszej części pracy, modele zawierające dane testowe dla aukcji jednostkowych wielu towarów, będące podstawą do tworzenia modeli zawierających dane testowe dla aukcji wielu jednostek wielu towarów, będą nazywane modelami bazowymi.

Pierwsze z przeprowadzonych badań modeli DMO-WDP dla aukcji wielu jednostek wielu towarów ma na celu sprawdzenie wpływu zmiany liczby jednostek poszczególnych towarów na czas wyznaczania rozwiązań przy stałej liczbie ofert, a rosnącej liczbie towarów. W tym celu wygenerowano 32 różne modele o rozkładzie *L5*. Każdy z tych modeli zawierał 1000 ofert, dla przypadków aukcji z kolejno wystawionymi liczbami towarów: 80, 160, 320, 640 po 5, 10, 20, 40, 80, 160, 320, 640 jednostek każdy. Ze względu na fakt, że wszystkie otrzymane wyniki z przeprowadzonych testów posiadają bardzo zbliżone czasy rozwiązań, zostaną one przedstawione na wspólnym rysunku 6.10.

Zmiana liczby jednostek poszczególnych towarów nie wpływa na czas wyznaczania rozwiązań w przypadkach aukcji, dla których liczba towarów wystawianych do sprzedaży jest większa od 160. W przypadkach, gdy liczba towarów wystawianych na aukcji jest mniejsza od 160, zmiany liczby jednostek tych towarów powodują zmiany czasu wyznaczania rozwiązań optymalnych. Zmiany te nie są proporcjonalne do wzrostu liczby jednostek towarów. Modele DMO-WDP zawierające teoretycznie łatwiejsze przypadki z mniejszą liczbą jednostek poszczególnych towarów powinny wyznaczać rozwiązania w krótszym czasie niż modele zawierające większą liczbę jednostek. Duże rozbieżności pomiędzy uzyskiwanymi czasami wyznaczania rozwiązań przy różnych liczbach jednostek towarów w przypadkach ofert zawierających do 80

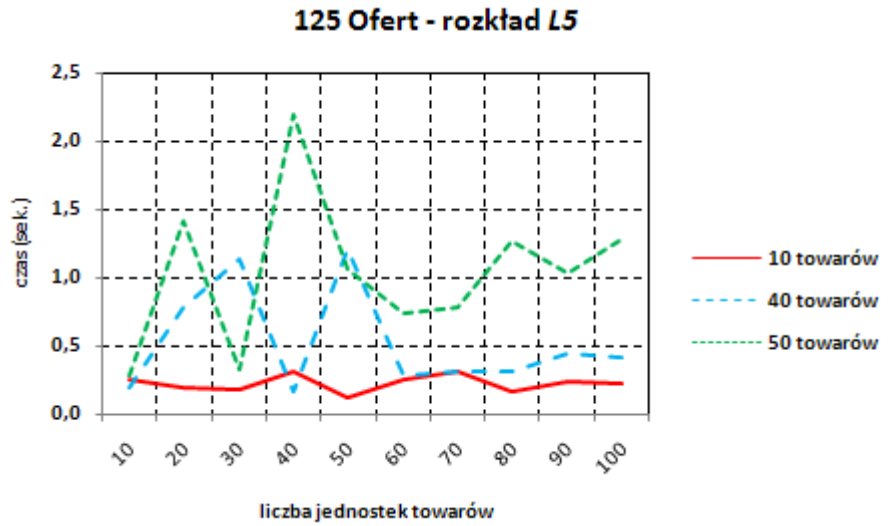


Rysunek 6.10: Modele DMO-WDP, rozkład ofert *L5*, aukcje wielu jednostek wielu towarów

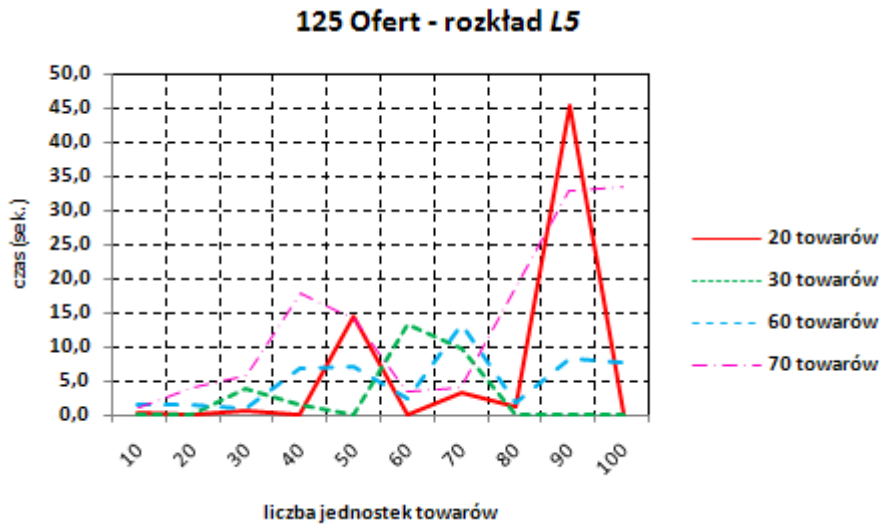
towarów, mogą zostać wyjaśnione różną efektywnością przeszukiwania drzewa decyzyjnego.

Ze względu na fakt, że przy dużej liczbie towarów wystawianych do sprzedaży, liczba jednostek tych towarów nie wpływa na czas wyznaczania rozwiązań, należy sprawdzić jak liczba jednostek wpływa na czas wyznaczania rozwiązań w przypadku małej liczby towarów. W tym celu przeprowadzono kolejne testy modeli DMO-WDP dla aukcji wielu jednostek wielu towarów. Testy te zostały przeprowadzone w oparciu o modele zawierające ponownie dane testowe o rozkładzie *L5*, lecz tym razem z mniejszą liczbą zgłoszonych ofert oraz mniejszą liczbą wystawionych na aukcji towarów. Ze względu na fakt, że testowane modele osiągały bardzo zróżnicowane czasy wyznaczania rozwiazań wyniki z tych testów, w celu lepszej ich czytelności, zostały przedstawione na dwóch rysunkach 6.11 oraz 6.12. Pierwszy z nich zawiera wykresy przebiegów czasowych modeli z małymi czasami wyznaczania rozwiązań optymalnych, natomiast drugi zawiera wykresy przebiegów czasowych z większymi czasami wyznaczania rozwiązań optymalnych.

Otrzymane wyniki testów wskazują, że nie można w przypadku małej liczby



Rysunek 6.11: Modele DMO-WDP, rozkład ofert *L5*, aukcje wielu jednostek wielu towarów



Rysunek 6.12: Modele DMO-WDP, rozkład ofert *L5*, aukcje wielu jednostek wielu towarów

towarów jednoznacznie określić wpływu liczby jednostek towarów na efektywność testowanych modeli. Wyjątkiem tutaj są przypadki modeli zawierające dane testowe dla aukcji z 10 wystawionymi do sprzedaży towarami. W tych przypadkach, gdy liczba jednostek poszczególnych towarów wynosiła 10, otrzymywane czasy wyznaczania rozwiazań okazały się niższe od czasów modeli bazowych dla aukcji jednostkowych wielu towarów. Wszystkie pozostałe wyniki wykazują duże zróżnicowanie w czasach wyznaczania rozwiązań w przypadkach wzrostu liczby towarów i liczby jednostek, jest to uzasadnione mechanizmem poszukiwań w drzewie decyzyjnym.

Reasumując: z przeprowadzonych testów modeli DMO-WDP dla aukcji wielu jednostek wielu towarów wynika, że w przypadku dużej liczby towarów (powyżej 160) wystawianych na aukcji, liczba jednostek tych towarów nie powoduje zmian czasu wyznaczania rozwiązań. Poniżej tej wartości modele wykazują bardzo duże zróżnicowanie w czasach wyznaczania rozwiazań. Nie ma natomiast reguły, która pozwoliłaby określić dokładną efektywność tych modeli w przedziale 20 - 160 towarów. Przeprowadzone testy pokazały, że im większa jest liczba towarów wystawianych na aukcji, tym czas wyznaczania rozwiązań jest krótszy. Modele CLP tym efektywniej rozwiązują zatem problem wyznaczenia zwycięzcy aukcji kombinatorycznej im większa jest liczba towarów wystawianych na aukcji, a więc im większa jest liczba ograniczeń opisujących daną aukcję. Efekt ten jest charakterystyczny dla rozwiązywania problemów optymalizacji kombinatorycznej metodą CLP: zwiększenie liczby ograniczeń z reguły skraca czas wyznaczania rozwiązania. W odróżnieniu, klasyczne metody optymalizacji całkowitoliczbowej wymagają zwiększenia liczby zmiennych decyzyjnych ze wzrostem liczby ograniczeń, co prowadzi do wydłużenia wyznaczania rozwiązania optymalnego. W tym momencie nasuwa się pytanie: czy dodatkowe addytywne ograniczenia komplementarne i/lub substytucyjne ofert również będą istotnie wpływały na czas wyznaczania rozwiązań?

6.7 Wpływ addytywnych ograniczeń na efektywność modeli

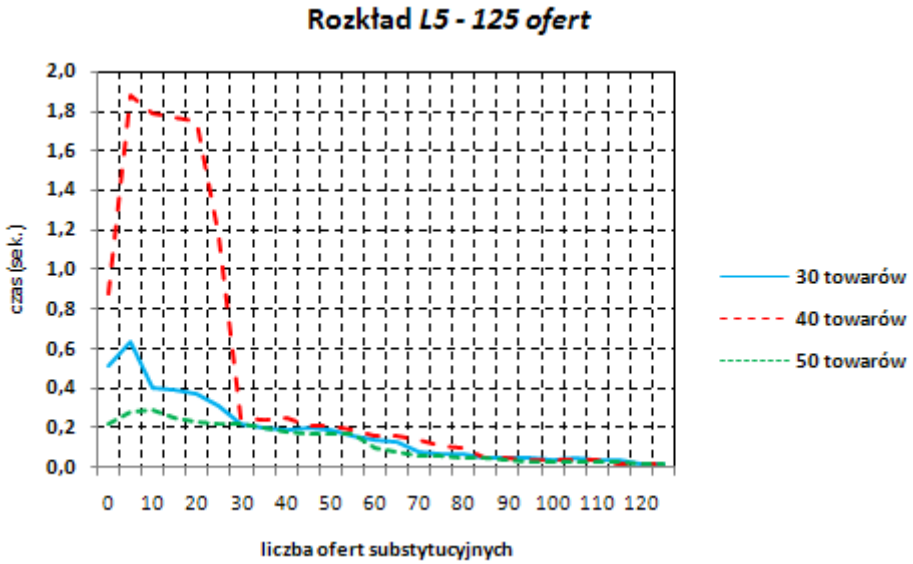
W przypadku, gdy reguły aukcji dopuszczają możliwość składania ofert komplementarnych i/lub substytucyjnych, fakt ten musi zostać odpowiednio zapisany w modelu CLP w postaci odpowiednio skonstruowanych ograniczeń, zwanych ograniczeniami addytywnymi. Ponieważ kluczowym elementem programowania w logice z ograniczeniami jest właśnie rozwiązywanie ograniczeń, dlatego należy sprawdzić w jaki sposób wpływają dodatkowe ograniczenia na efektywność prezentowanych w pracy modeli.

6.7.1 Ograniczenia substytucyjne

Aby określić wpływ dodatkowych ograniczeń substytucyjnych na efektywność modeli DMO-WDP, zostały przeprowadzone kolejne testy. Pierwszy z nich dotyczy modelu zawierającego dane testowe o rozkładzie *L5*, zawierającego stałą liczbę ofert (125) oraz następujące liczby towarów wystawionych do sprzedaży - 30, 40, 50. W modele te implementowano ograniczenia substytucyjne ofert w następujący sposób:

- w pierwszym kroku sprawdzono czas wyznaczenia rozwiązania optymalnego bez dodatkowych ograniczeń substytucyjnych,
- w każdym następnym kroku ponownie sprawdzano czasy wyznaczania rozwiązań optymalnych, za każdym razem dodając jedno ograniczenie substytucyjne, mówiące, że pięć kolejnych ofert jest substytucyjnie ograniczonych.

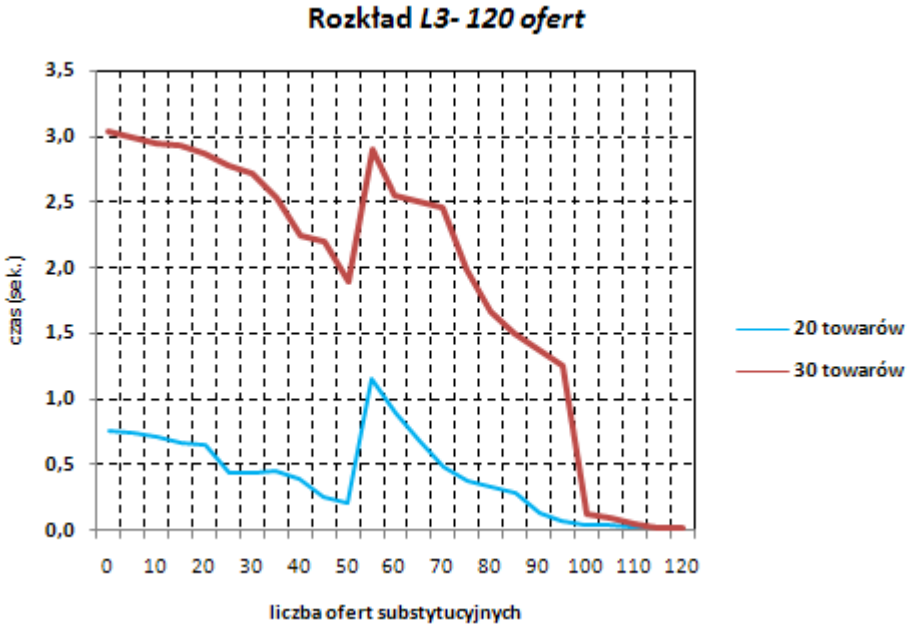
Dodawanie ograniczeń substytucyjnych do modelu CLP trwało aż do momentu ograniczenia substytucyjnego wszystkich ofert. Ograniczenie substytucyjne wszystkich ofert oznacza w tym przypadku, że testowane modele zawierały 125 ofert substytucyjnych, złożonych przez 25 oferentów po 5 ofert każdy. Otrzymane wyniki przedstawione zostały w postaci trzech wykresów na rysunku 6.13.
Po wykonaniu analizy otrzymanych wyników można jednoznacznie powiedzieć, że w przypadku danych testowych o rozkładzie *L5* niewielka liczba ofert substytucyjnych (poniżej 5%wszystkich złożonych ofert) powoduje wzrost czasu wyznaczania rozwiązania optymalnego. Wzrost ten uzależniony jest od



Rysunek 6.13: Implementacja ograniczeń substytucyjnych w modele DMO-WDP, rozkład ofert *L5*

liczby towarów. Im większa jest liczba towarów, tym wzrost czasu wyznaczania rozwiązania optymalnego jest większy. Badane ograniczenia substytucyjne implementowane w model CLP, w przypadku gdy obejmują powyżej 5% wszystkich złożonych ofert, powodują skrócenie czasu wyznaczania rozwiązania optymalnego. Po umieszczeniu w modelu CLP ograniczeń substytucyjnych dla około 30% wszystkich ofert, czas wyznaczania rozwiąznia optymalnego był niższy od czasu wyznaczania rozwiązań optymalnych przez modele CLP nie zawierające ograniczeń substytucyjnych.

Aby dokładniej sprawdzić wpływ umieszczania dodatkowych ograniczeń substytucyjnych w modelach DMO-WDP na ich efektywność, przeprowadzone zostały kolejne testy. Do tego celu wygenerowane zostały dane testowe o rozkładzie *L3*. Testom poddano modele zawierające 120 ofert i 20 oraz 30 towarów wystawionych do sprzedaży. Do modeli tych podobnie jak podczas testowania modeli o rozkładzie *L3* dodawane zostały kolejne ograniczenia substytucyjne, aż do momentu gdy wszystkie oferty zostały objęte tymi ograniczeniami. Wyniki z przeprowadzonych testów przedstawione zostały na rysunku 6.14.



Rysunek 6.14: Implementacja ograniczeń substytucyjnych w modele DMO-WDP, rozkład ofert *L3*

W przypadku modeli zawierających dane testowe o rozkładzie *L3* okazało się, że w momencie dodania do modelu pierwszego ograniczenia substytucyjnego, czas wyznaczania rozwiązania zmniejszył się. Tendencja spadkowa czasu wyznaczania rozwiazań trwa do momentu, gdy ograniczeniami substytucyjnymi jest objętych 50 z 120 ofert. Gwałtowny wzrost czasu wyznaczania rozwiazań optymalnych występuje w przypadku, gdy ograniczeniami substytucyjnymi jest objęte 55 ofert. Po dodaniu kolejnych ograniczeń, ponownie występuje tendencja spadkowa czasu, aż do osiagnięcia swojego minimum w momencie, gdy ograniczeniami substytucyjnymi objęte są wszystkie oferty.

Po przeprowadzeniu szeregu testów dotyczących implementacji dodatkowych ograniczeń substytucyjnych w modele DMO-WDP zauważyć można, że ograniczenia te istotnie wpływają na czas wyznaczania rozwiazań optymalnych. W większości przypadków implementacja dodatkowych ograniczeń substytucyjnych w modele CLP powoduje zmniejszenie czasu wyznaczania rozwiązań,

jednakże istnieją przypadki, dla których implementacja tych ograniczeń powoduje wzrost czasu wyznaczania rozwiązań. Wszystkie umieszczane w testowanych modelach dodatkowe ograniczenia substytucyjne ofert mogą nie odzwierciedlać ograniczeń substytucyjnych, występujących w realnych warunkach.

6.7.2 Ograniczenia komplementarne

W celu określenia wpływu ograniczeń komplementarnych na efektywność modeli CLP należałoby przeprowadzić kolejne testy dla tych modeli. Jednakże zaproponowany sposób modelowania ograniczeń komplementarnych, a wiec łączenie ofert komplementarnych i deklarowanie ich jako jednej, pozwala na określenie wpływu tych czynności na efektywność modeli bez konieczności wykonywania testów. Łączenie wszystkich ofert komplementarnych, złożonych przez tego samego oferenta, powoduje zmniejszenie rozmiaru problemu, a więc im więcej ofert komplementarnych zostaje połączonych, tym bardziej zmniejsza się liczba wszystkich składanych na aukcji ofert. Wniosek zatem jest jeden: wraz ze zmniejszeniem rozmiaru problemu skraca się czas wyznaczania rozwiązania lub rozwiązań optymalnych.

Rozdział 7

Podsumowanie i wnioski końcowe

7.1 Cel pracy

Celem głównym niniejszej pracy było opracowanie efektywnych modeli, metod poszukiwania i optymalizacji dla ogólnego problemu wyznaczenia zwycięzcy aukcji kombinatorycznej za pomocą paradygmatu programowania w logice z ograniczeniami. W niniejszej pracy cel ten został osiągnięty poprzez budowę dwóch modeli CLP, różniących się metodami modelowania ograniczeń, występujących w problemie WDP, a pozwalających efektywnie rozwiązywać problem WDP. Przeprowadzone i opisane w pracy testy pozwoliły określić najefektywniejsze parametry tych modeli. Parametry te determinują metody poszukiwania rozwiązań oraz metody optymalizacji tych rozwiązań. Cele cząstkowe, składające się na cel główny, a więc:

- zbadanie efektywności czasowej tych modeli,
- zbadanie wpływu zmienności danych na efektywność czasową modeli,
- analiza wpływu dodatkowych ograniczeń w modelach CLP na czas poszukiwania rozwiązań problemu WDP,
- określenie wpływu zmian strategii numeracyjnych na wielkość przestrzeni decyzyjnej oraz na czas poszukiwania rozwiązań problemu WDP,
- określenie wpływu zmiany metody ukonkretniania zmiennych, na czas wyznaczania rozwiązania optymalnego,

zostały osiągnięte za pomocą przeprowadzonych badań oraz otrzymanych wyników opisanych szczegółowo w punkcie 6.4 niniejszej pracy.

7.2 Słuszność postawionych w pracy tez

Postawione na wstępie pracy tezy zostały w niej wykazane:

- Za pomocą techniki programowania w logice z ograniczeniami można budować modele, umożliwiające efektywne rozwiązywanie problemu wyznaczenia zwycięzcy aukcji kombinatorycznej. Zaprezentowane w pracy modele CLP (SMO-WDP, DMO-WDP) jednoznacznie pokazują, że metoda CLP może być z powodzeniem wykorzystywana do modelowania i rozwiązywania problemu WDP dla różnych typów aukcji kombinatorycznych, zarówno jednostkowych wielu towarów, jak i dla wielu jednostek wielu towarów.

- Metoda programowania w logice z ograniczeniami jest na tyle elastyczna, że umożliwia tworzenie modeli CLP dla problemu WDP na wiele różnych sposobów. W pracy zostały opisane ostatnie wersje najbardziej efektywnych modeli, jednakże podczas dochodzenia do ostatecznych wersji tych modeli zostało napisanych wiele wersji, różniących się sposobem modelowania zarówno danych wejściowych jak i ograniczeń, a także późniejszym sposobem przetwarzania tych danych.

- Zaprezentowane w pracy modele, pozwalają na łatwą implementację ograczniczeń zarówno ogólnych jak i addytywnych (substytucyjnych i/lub komplementarnych). Ograniczenia te mają istotny wpływ na czas wyznaczania rozwiązania lub rozwiązań optymalych, co zostało potwierdzone wynikami przeprowadzonych testów.

- Znaczący wpływ na czas wyznaczania rozwiązań dopuszczalnych i/lub rozwiązań optymalnych ma wybór metody przeszukiwania drzew rozwiązań. Wybór odpowiedniej metody pozwala na zmniejszenie przeszukiwanej przestrzeni decyzyjnej.

- Czas wyznaczenia rozwiązania lub rozwiązań optymalnych dla modeli CLP nie jest monotoniczną funkcją liczby towarów wystawianych na aukcji.

- Pomiędzy sformułowaniem problemu za pomocą programowania w logice z ograniczeniami, a programem rozwiązującym problem wyznaczania zwycięzcy aukcji kombinatorycznej nie ma przepaści semantycznej. W przypadku modeli SMO-WDP słowny zapis sformułowania problemu jest programem go rozwiązującym. Natomiast w przypadku modeli DMO-WDP, zapis ten jest tworzony dynamicznie za pomocą odpowiednich predykatów.

7.3 Wnioski z przeprowadzonych analiz i testów

7.3.1 Programowanie w logice z ograniczeniami

Programowanie w logice z ograniczeniami pozwala, dzięki wbudowanym mechanizmom, na znaczne ograniczenie przestrzeni decyzyjnej problemu. Można to uzyskać między innymi za pomocą różnych stategii numeracyjnych, które wpływają na miejsca rozpoczęcia i sposób przeglądania drzewa decyzyjnego, oraz za pomocą różnych metod ukonkretniania zmiennych. W prezentowanych modelach CLP można wprowadzać modyfikacje tych elementów za pomocą odpowiednich parametrów opisanych w punkcie 6.4.

Programowanie w logice z ograniczeniami pozwala modelować występujące w problemie WDP ograniczenia na różne sposoby. Może się to odbywać statycznie lub też dynamicznie. Statyczne modelowanie ograniczeń to ręczne wpisywanie w model matematycznych zależności, zawartych pomiędzy ofertami. Dynamiczne modelowanie ograniczeń odbywa się natomiast za pomocą specjalnie napisanych predykatów własnych, budujących dynamicznie te ograniczenia podczas działania programu.

Ograniczenie maksymalnej wartości, jaką mogą przyjmować domeny zmiennych w oprogramowaniu CHIP v5.8, okazało się podstawową niedogodnością tego oprogramowania podczas budowy modeli CLP dla problemu WDP. Niedogodność ta związana jest z faktem, że modelowane przypadki aukcji mogą posiadać wartości cenowe ofert, przekraczające maksymalną wartość jaką może przyjmować domena zmiennych. W rzeczywistych warunkach zgłaszane na aukcjach oferty zawierają ceny szacowane niejednokrotnie w milionach. Zaproponowane w pracy modele pozwalają na wyznaczenie rozwiązań takich aukcji, jednakże dopiero po odpowiednim przeskalowaniu

wartości cen wszystkich zgłoszonych ofert.

7.3.2 Modele SMO-WDP i DMO-WDP

Przedstawione w pracy dwa modele CLP, różniące się metodami modelowania ograniczeń i funkcji celu, posiadają taką samą efektywność czasową, co zostało poparte wynikami odpowiednich testów. Dodatkowo testy te pokazały, że metoda modelowania ograniczeń i funkcji celu nie wpływa na liczbę generowanych rozwiązań dopuszczalnych oraz na końcowe rozwiązanie lub rozwiązania optymalne.

Oba modele umożliwiają efektywne rozwiązywanie problemu WDP zarówno dla aukcji kombinatorycznych jednostkowych wielu towarów jak i wielu jednostek wielu towarów. W przypadku modelu SMO-WDP zmiana typu aukcji wiąże się ze zmianą danych wejściowych, a więc z całkowitą przebudową wszystkich występujących w danym przypadku ograniczeń i funkcji celu. Model DMO-WDP natomiast nie wymaga wprowadzania żadnych modyfikacji podczas zmiany typu aukcji, gdyż wyznacza on rozwiązania na podstawie danych wejściowych, które zawsze zawierają informacje o liczbach jednostek danego towaru.

Prezentowane w pracy modele CLP, pozwalają niemal natychmiastowo po ich uruchomieniu udzielić odpowiedzi na pytanie, czy jest możliwa alokacja wszystkich wystawionych na aukcji towarów w otrzymanych ofertach ich kupna. Oczywiście rozwiązanie to nie musi być optymalne pod względem finansowym, jednakże w przypadku chęci sprzedaży wszystkich towarów przez aukcjonera (nie patrząc na efekt finansowy), za pomocą prezentowanych modeli jest łatwo sprawdzalne.

Optymalność otrzymanych rozwiązań wynika całkowicie z optymalności gwarantowanej przez predykat *min_max*, realizujący metodę *Branch and Bound*. Optymalność ta została wielokrotnie potwierdzona dla niezbyt dużych przykładów, metodą przeglądu zupełnego.

7.3.3 Addytywne ograniczenia

Implementacja dodatkowych ograniczeń ma istotny wpływ na czas wyznaczania rozwiązania lub rozwiązań optymalnych. Dodatkowe ograniczenia substytucyjne oraz ograniczenia komplementarne (łączenie ofert komplementarnych) powodują zmniejszanie się drzewa decyzyjnego, a co za tym idzie skrócenie czasu wyznaczania rozwiązań.

Implementacja w modelach DMO-WDP dodatkowych addytywnych ograniczeń substytucyjnych i/lub komplementarnych nie powoduje pogorszenia czytelności tych modeli, natomiast implementacja tych samych ograniczeń w modelach SMO-WDP powoduje znaczne pogorszenie czytelności tych modeli. Modele DMO-WDP pozwalają na implementację dodatkowych ograniczeń substytucyjnych za pomocą predykatów, budujących ograniczenia ogólne.

Dodatek A

Praktyczna implementacja modeli

W rozdziale tym przedstawiono internetowy system, wspomagający przeprowadzenie dowolnej aukcji kombinatorycznej. W systemie tym zaimplementowane zostały modele opisane w rozdziale 5 niniejszej pracy.

A.1 Założenia systemu

Internetowy system wspomagający przeprowadzenie dowolnej aukcji kombinatorycznej powinien spełniać kilka podstawowych założeń:

- rozgraniczać dostęp do systemu dla administratora (aukcjonera zarządzającego systemem) oraz oferentów (klientów systemu),
- umożliwiać aukcjonerowi pełne zarządzanie aukcjami, w tym tworzenie nowych aukcji oraz edycję już stworzonych,
- umożliwić zgłaszanie ofert kupna oferentom za pomocą internetu,
- system powienien posiadać funkcje importu danych z odpowiednio sformatowanych plików tekstowych oraz eksportu do pliku tekstowego, zawierającego dane oraz odpowiedni model CLP,
- po zakończeniu aukcji wyznaczyć oferty wygrywające i udostępnić te wyniki w internecie.

A.2 Wykorzystane narzędzia

Do stworzenia internetowego systemu wspomagającego przeprowadzenie aukcji kombinatorycznej zostały wykorzystane następujące narzędzia:

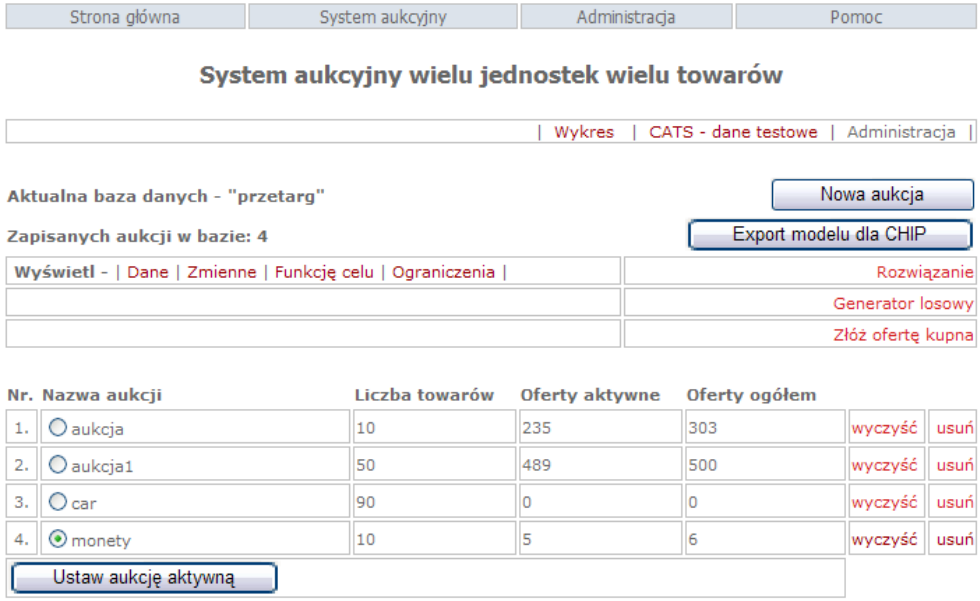
- **MySQL** - relacyjna baza danych, w której przechowywane są informacje, dotyczące budowy i struktury aukcji oraz późniejsze, napływające dane w postaci ofert,
- **HTML, CSS, PHP** - języki programowania, które umożliwiły napisanie wszystkich interfejsów komunikacyjnych systemu,
- **CHIP v5.8** - oprogramowanie wykorzystane do automatycznego rozwiązania problemu wyznaczenia ofert wygrywających (problem WDP), na podstawie modeli SMO-WDP i DMO-WDP wygenerowanych przez funkcję eksportu systemu internetowego.

A.3 Internetowy system aukcji kombinatorycznych

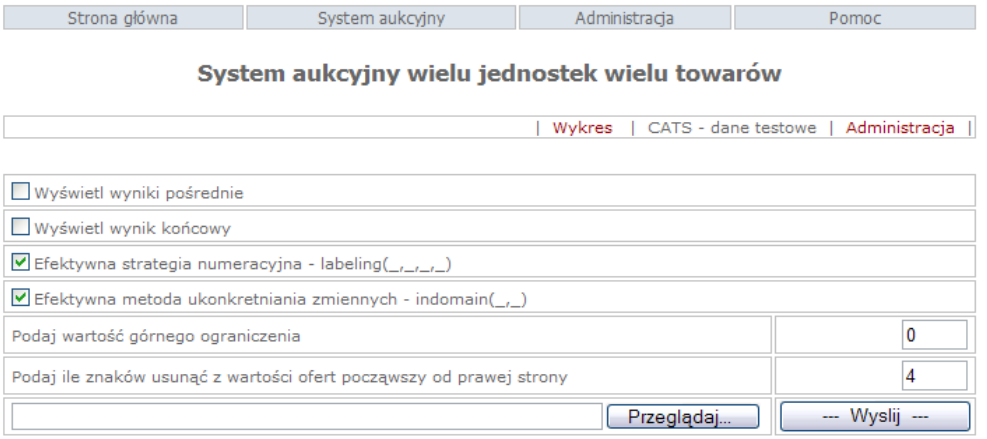
Internetowy system wspomagający przeprowadzenie aukcji kombinatorycznej można obsługiwać za pomocą dowolnej przeglądarki internetowej. Umożliwia on przeprowadzanie dowolnej aukcji kombinatorycznej jednostkowej wielu towarów oraz dowolnej aukcji kombinatorycznej wielu jednostek wielu towarów, w zależności od liczby deklarowanych towarów podczas tworzenia aukcji. System składa się z dwóch zależnych modułów. Pierszy z nich to moduł zarządzający, obsługiwany przez aukcjonera. Drugi to moduł kliencki, pozwalający na rejestrację zgłaszanych ofert kupna. Wyznaczenie ofert wygrywających odbywa się za pomocą kompilatora CHIP na podstawie eksportowanego z modułu zarządzającego pliku tekstowego, zawierającego kompletny kod źródłowy modelu DMO-WDP wraz z danymi zebranymi podczas trwania aukcji.

Internetowy system aukcji komninatorycznych pozwala oprócz przeprowadzenia realnych aukcji również na przetwarzanie danych testowych z programu *CATS* 6.2. Po zaimportowaniu do systemu odpowiednio sformatowanego pliku tektowego, zawierającego dane testowe wygenerowane z programu CATS, system umożliwia wygenerowanie na podstawie tych danych modelu

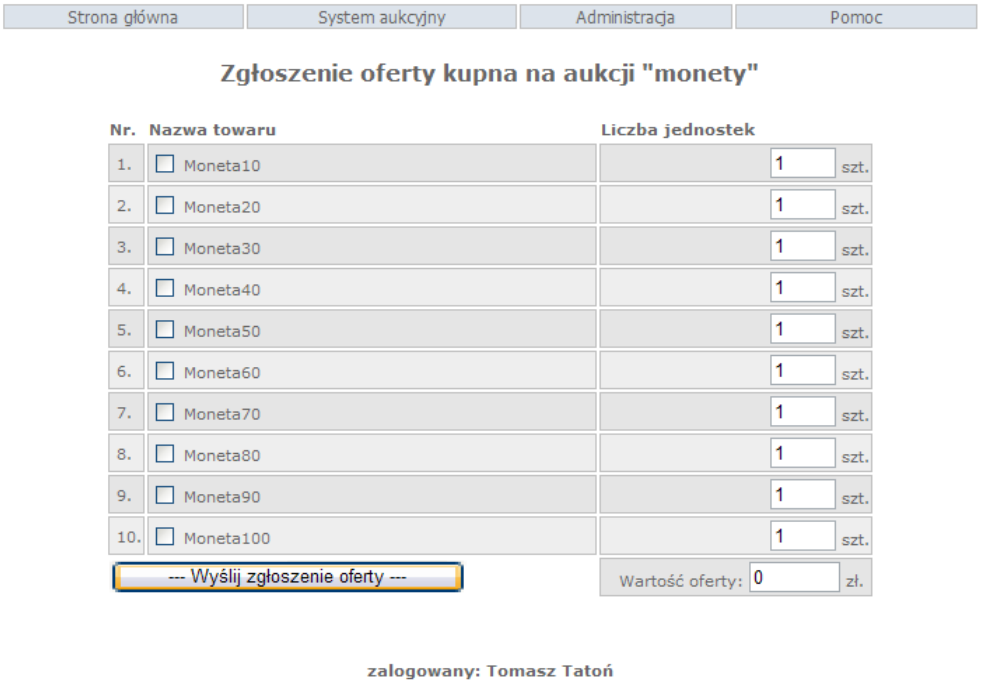
DMO-WDP. Podczas tworzenia przez system modeli CLP dla problemu WDP można zmieniać jego parametry pracy opisane w punkcie 6.4.



Rysunek A.1: Internetowy system wspomagający przeprowadzenie aukcji kombinatorycznej - moduł administracyjny



Rysunek A.2: Internetowy system wspomagający przeprowadzenie aukcji kombinatorycznej - moduł administracyjny



Rysunek A.3: Internetowy system wspomagający przeprowadzenie aukcji kombinatorycznej - moduł kliencki

Dodatek B

Stosowane oznaczenia

a_{ij}	liczba jednostek towaru i znajdująca się w ofercie j
B	zbiór wszystkich złożonych na aukcji ofert
B_j	j-ta oferta złożona na aukcji (j-ta para składająca się z S_j i p_j)
CLP	Programowanie w logice z ograniczeniami
$DMO-WDP$	Model CLP dla problemu WDP bazujący na dynamicznie budowanych ograniczeniach
$_i$	należy do zbioru od 1 do m (indeks towaru wystawionego na aukcji)
$_j$	należy do zbioru od 1 do n (indeks uczestnika aukcji,a więc indeks złożonej przez niego oferty)
K	liczba wszystkich ofert komplementarnych, złożonych przez tego samego oferenta
M	zbiór wszystkich towarów wystawionych na aukcji do sprzedaży
M_i	i-ty towar wystawionych na aukcji do sprzedaży
m	liczba towarów wystawionych na aukcji do sprzedaży

n	liczba uczestników aukcji (liczba złożonych ofert)
p_j	cena, którą j-ty uczestnik aukcji jest skłonny zapłacić za S_j
S	liczba wszystkich ofert substytucyjnych, złożonych przez tego samego oferenta
$_s$	należy do zbioru od 1 do S (indeks oferenta, składającego oferty substytucyjne i/lub komplementarne)
S_j	j-ty podzbiór zbioru M
$SMO-WDP$	Model CLP dla problemu WDP, bazujący na statycznie wpisywanych ograniczeniach
U	zbiór wszystkich jednostek dla wszystkich towarów wystawionych na aukcji
U_i	liczba jednostek i-tego towaru wystawionego na aukcji
x_j	binarna zmienna decyzyjna j-tej oferty (ze względu na wymagania oprogramowania CHIP v5.8 zmienna ta występuje w kodach źródłowych modeli jako Xj)
x_{js}	binarna zmienna decyzyjna j-tej oferty substytucyjnej i/lub komplementarnej s-tego oferenta
WDP	Problem wyznaczenia zwycięzcy aukcji kombinatorycznej

Dodatek C

Kody źródłowe modeli CLP

C.1 Kod źródłowy modelu SMO-WDP

```
?- wflags(128).
% Ofert -  10
% Towarow -  5

top:-
     utime(_),
     Decyzje=[X1,X2,X3,X4,X5,X6,X7,X8,X9,X10],
     Decyzje::0..1,
     Temp :: 0..100000,
     Zysk :: 0..100000,

     X2+X4+X5+X8#<=1,
     X4+X6+X9+X10#<=1,
     X3+X5+X6#<=1,
     X1+X2+X6+X7+X9#<=1,
     X1+X3+X8#<=1,

     Zysk #=X1*98+X2*74+X3*97+X4*47+X5*69+X6*79+X7*36+
            X8*86+X9*67+X10*41,
     Temp + Zysk#= 100000,

     min_max((labeling(Decyzje,0,most_constrained,fix),
             get_choice(C)),Temp),

     writeln(Decyzje),
     write(' Wartosc przyjetych ofert = '),
```

```
     writeln(Zysk),
     write(' Rozwiazanie optymalne wyznaczono w czasie = '),
     writeln_time,
     open('wynik.txt',S,w),
     write(S,Decyzje),
     close(S).

get_choice(C):-
     getval(choice,N),
     setval(time,N),
     writeln(N),
     inc_choice.

inc_choice:-
     incval(choice,_),
     fail.
inc_choice.

fix(X):-
     indomain(X,max).
```

C.2 Kod źródłowy modelu DMO-WDP

```
?- wflags(128).
% Ofert -  10
% Towarow -  5

ceny([98,74,97,47,69,79,36,86,67,41]).

specyfikacje([
     [0,1,0,1,1,0,0,1,0,0],
     [0,0,0,1,0,1,0,0,1,1],
     [0,0,1,0,1,1,0,0,0,0],
     [1,1,0,0,0,1,1,0,1,0],
     [1,0,1,0,0,0,0,1,0,0]
     ]).

jednostki([1,1,1,1,1]).

tworz_liniowe([],[],0).
```

```
tworz_liniowe([A|Li],[X|Vars],A*X+Liniowe_wyrazenie) :-
     !,
     tworz_liniowe(Li,Vars,Liniowe_wyrazenie).

tworz_ograniczenie_ogolne_sub([],Vars,[]).

tworz_ograniczenie_ogolne_sub([],_,_) :-
     sprintf(Msg,"Blad: rozna dlugosc list",[]),
     writeln(Msg),
     fail.

tworz_ograniczenie_ogolne_sub(_,_,[]) :-
     sprintf(Msg,"Blad: rozna dlugosc list",[]),
     writeln(Msg),
     fail.

tworz_ograniczenie_ogolne_sub([Li|List],Vars,[Ui|Jednostki]) :-
     !,
     tworz_liniowe(Li,Vars,Liniowe_wyrazenie),
     Liniowe_wyrazenie #<= Ui,
     tworz_ograniczenie_ogolne_sub(List,Vars,Jednostki).

top :-
     length(Decyzje,10),
     utime(_),
     Decyzje :: 0..1,
     Temp :: 0..100000,
     Zysk :: 0..100000,

     ceny(Ceny),
     specyfikacje(Dane),
     jednostki(Jednostki),

     tworz_ograniczenie_ogolne_sub(Dane,Decyzje,Jednostki),
     tworz_liniowe(Ceny,Decyzje,Liniowe_wyrazenie),

     Liniowe_wyrazenie #= Zysk,
     Temp + Zysk  #= 100000,

     min_max((labeling(Decyzje,0,most_constrained,fix),
              get_choice(C)),Temp),
```

```
     writeln(Decyzje),
     write(' Wartosc przyjetych ofert = '),
     writeln(Zysk),
     write(' Rozwiazanie optymalne wyznaczono w czasie = '),
     writeln_time,
     open('wynik.txt',S,w),
     write(S,Decyzje),
     close(S).

get_choice(C):-
     getval(choice,N),
     setval(time,N),
     writeln(N),
     inc_choice.

inc_choice:-
     incval(choice,_),
     fail.

inc_choice.

fix(X):-
 indomain(X,max).
```

Bibliografia

[And03] A. Andersson. Combinatorial auctions, an example of algorithm theory in real life. *Lecture Notes in Computer Science*, 2598:13–21, 2003.

[Apt03] K. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003.

[ATY00] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 39–46, 2000.

[Bar98] R. Bartâk. On-line guide to constraint programming, 1998. `http://kti.mff.cuni.cz/~bartak/constraints/`, (strona dostępna w dniu, 1.05.2008).

[Bar99] R. Bartâk. Constraint programming: What is behind? In *Proceedings of the CPDC'99 Workshop on Constraint Programming for Decision and Control*, pages 7–15, 1999.

[Bar01] R. Bartâk. Dynamic global constraints in constraint logic programming. ITI Series 2001-018, 2001. `http://kti.mff.cuni.cz/~bartak/html/publications.html`, (strona dostępna w dniu, 1.05.2008).

[BL06] S. Bouveret and M. Lemaître. Finding leximin-optimal solutions using constraint programming: new algorithms and their application to combinatorial auctions, 2006. `http://staff.science.uva.nl/~ulle/COMSOC-2006/papers/30-bouveret.pdf`, (strona dostępna w dniu, 1.05.2008).

[BU01] C. Baral and C. Uyan. Declarative specification and solution of combinatorial auctions using logic programming. *Lecture Notes in Computer Science*, 2173:186–199, 2001.

[Cla71] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11, 1971.

[CM03] W.F. Clocksin and C.S. Mellish. *Prolog programowanie*. Wydawnictwo Helion, 2003.

[Col] Alain Colmerauer. *An Introduction to Prolog III*. `http://alain.colmerauer.free.fr/ArchivesPublications/PrologIII/acmprolog3e.pdf`, (strona dostępna w dniu, 1.05.2008).

[Con] Cosytec Consortium. *CHIP v5.7 Programming System*. Documentation and system available via WWW from `http://www.cosytec.com`, (strona dostępna w dniu, 1.05.2008).

[CSe06] P. Cramton, Y. Shoham, and R. Steinberg (eds.). *Combinatorial Auctions*. MIT PRESS, 2006.

[Dia] Daniel Diaz. *The GNU Prolog web site*. Documentation and system available via WWW from `http://www.gprolog.org/`, (strona dostępna w dniu, 1.05.2008).

[FLBS99] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *International Joint Conferences on Artificial Intelligence*, pages 548–553, 1999.

[FT05] K. Fleszar and E. Toczyłowski. Algorytmy przybliżonego rozwiązywania problemu aukcji kombinatorycznej. In *Krajowa konferencja automatyki*, volume XV, pages 335–340, 2005.

[Gal] A. Galen. *CATS 2.0 User Guide*. Documentation and system available via WWW from `http://www.cs.ubc.ca/~kevinlb/CATS/`, (strona dostępna w dniu, 1.05.2008).

[GL00] R. Gonen and D. Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics, 2000. `http://www.cs.huji.ac.il/~rgonen/`, (strona dostępna w dniu, 1.05.2008).

[GL01] R. Gonen and D. Lehmann. Linear programming helps solving large multi-unit combinatorial auction, 2001. `http://www.cs.huji.ac.il/~rgonen/papers/RGonenLP.ps`, (strona dostępna w dniu, 1.05.2008).

[GS91] E. Gatnar and K. Stąpor. *Prolog*. Wydawnictwo PLJ, 1991.

[GSL06] C. Gallo, G. De Stasio, and C. Di Letizia. A data set generation algorithm in combinatorial auctions. Technical report, Dipartimento di Scienze Economiche, Matematiche e Statistiche, Universita' di Foggia, 2006.

[GSS93] R.L. Graves, L. Schrage, and J.K. Sankaran. An auction method for course registration. *Interfaces*, 23(5):81–92, 1993.

[Han03] J. Hansen. Constraint programming versus mathematical programming. *Journal paper, Orbit*, 2003.

[Har01] D. Harel. *Rzecz o istocie informatyki Algorytmika*. Wydawnictwo Naukowo Techniczne Warszawa wydanieIII, 2001.

[HB00] H. Hoos and C. Boutilier. Solving combinatorial auctions using stochastic local search. In *Proceedings American Association for Artificial Intelligence*, pages 22–29. MIT Press, 2000.

[HM01] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, pages 449–467, 2001.

[HO04] A. Holland and B. O'Sullivan. Super solutions for combinatorial auctions. *ERCIM-Colognet Constraints Workshop (CSCLP 04)*, 3419:187–200, 2004.

[Hol01] R. C. Holte. Combinatorial auctions, knapsack problems, and hill-climbing search. *Lecture Notes in Computer Science*, 2056:57–66, 2001.

[JM94] J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[Kel04] T. Kelly. Generalized knapsack solvers for multi-unit combinatorial auctions: Analysis and application to computational resource allocation. Technical report, HP Laboratories Palo Alto, HPL-2004-21, 2004.

[LBPS00] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms, 2000. `http://robotics.stanford.edu/~kevinlb/CATS.pdf`, (strona dostępna w dniu, 1.05.2008).

[LBS06] K. Leyton-Brown and Y. Shoham. *A Test Suite for Combinatorial Auctions*, chapter Testing and Implementation, pages 451–477. MIT Press, 2006.

[LBST00] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *AAAI/IAAI*, pages 56–61, 2000.

[Leg06] W. Legierski. *Automated Timetabling via Constraint Programming*. PhD thesis, Silesian University of Technology, 2006.

[LMS06] D. Lehmann, R. Müller, and T. Sandholm. *The Winner Determination Problem*, chapter Complexity and Algorithmic Considerations, pages 297–317. MIT PRESS, 2006.

[MB06] T. Munakata and R. Bartâk. Combinatorics in logic programming: Implementations and applications. *International Journal of Information Technology and Intelligent Computing*, Vol.1, No.2, IEEE Computational Intelligence Society:419–428, 2006.

[McM94] J. McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, 8(3):145–162, 1994.

[Mon] Eric Monfroy. *Getting started with CLP(R).* `http://www.sju.edu/~jhodgson/clp/howto_clpr.html`, (strona dostępna w dniu, 1.05.2008).

[ND05] Y. Narahari and P. Dayama. Combinatorial auctions for electronic business. *Sadhana Bangalore*, volume 30, Part. 2/3:179–211, 2005.

[Nie99] A. Niederliński. Constraint logic programming - from Prolog to CHIP. In *Proceedings of the CPDC'99 Workshop on Constraint Programming for Decision and Control*, pages 27–34, 1999.

[Nie01] A. Niederliński. Making backtracking and Branch-and-Bound more effective. In *Proceedings of the CPDC'01 Workshop on Constraint Programming for Decision and Control*, pages 47–52, 2001.

[Nie06] A. Niederliński. *Regułowo - modelowe systemy ekspertowe rmse*. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, 2006.

[Nis00] N. Nisan. Bidding and allocation in combinatorial auctions, 2000. `http://www.cs.huji.ac.il/~noam/mkts.html`, (strona dostępna w dniu, 1.05.2008).

[NS97] I. Niemela and P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic program. In *Proc. 4th international conference on Logic programming and non-monotonic reasoning*, pages 420–429, 1997.

[Qua94] D.C. Quan. Real estate auctions: A survey of theory and practice. *The Journal of Real Estate Finance and Economics*, 9(1):23–49, 1994.

[RPH98] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[RvBW06] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[San02] T Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[Sch03] L. Schrage. Solving multi-object auctions with lp/ip. *Seminar series: Department of Quantitative Analysis and Operations Management College of Business Administration University of Cincinnati*, 2003.

[SS03] T. Sandholm and S. Suri. Bob: Improved winner determination in combinatorial auctions and generalizations. *Artificial Intelligence*, 145:33–58, 2003.

[SSGL01a] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: a fast optimal algorithm for winner determination in combinatorial auctions. In *IJCAI*, pages 1102–1108, 2001.

[SSGL01b] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. *In AGENTS-2001 Workshop on Agent-Based Approaches to B2B, Montreal, Canada*, 2001.

[SWB05] P. Sitek, J. Wikarek, and Z. Banaszak. Clp - based approach to the job shop scheduling problem (JSSP). In *Proceedings of the 7 ĥ Workshop on Constraint Programming for Decision and Control*, pages 21–26, 2005.

[SY04] Jin-Woo Song and Sung-Bong Yang. A faster optimal allocation algorithm in combinatorial auctions. *Lecture Notes in Computer Science*, 2972:362–369, 2004.

[Sys] Cisco Systems. *The ECL^iPS^e web site.* Documentation and system available via WWW from `http://eclipse-clp.org/`, (strona dostępna w dniu, 1.05.2008).

[Szc02] T. Szczygieł. Angle packing problems: Traditional approaches vs. constraint logic programming approaches. In *Proceedings of the CPDC'99 Workshop on Constraint Programming for Decision and Control*, pages 67–73, 2002.

[Szk07] R. Szklarczyk. Constraint logic programming solution for capacitated vehicle routing problem. In *Recent Developments in Artificial Intelligence Methods*, pages 215–222. AI-METH Series, 2007.

[Tat06] T. Tatoń. Zastosowanie programowania ograniczeniowego w logice dla problemu konfigurowania zespołów pracowników. In *Zeszyty Naukowe Politechniki Śląskiej z. 145*, volume XV of *Krajowa Konferencja Automatyzacji Procesów Dyskretnych*, pages 209–216. Wydawnictwo Politechniki Śląskiej, 2006.

[Tat07] T. Tatoń. A constraint logic programming approach to the winner determination problem. In *Recent Developments in Artificial Intelligence Methods*, pages 231–239. AI-METH Series, 2007.

[Toc02] E. Toczyłowski. *Optymalizacja procesów rynkowych przy ograniczeniach.* Akademicka Oficyna Wydawnicza EXIT, 2002.

[Trz03] T. Trzaskalik. *Wprowadzenie do badań operacyjnych z komputerem.* Polskie Wydawnictwo Ekonomiczne, Warszawa, 2003.

[TWW+01] TAC Team, M.P. Wellman, P.R. Wurman, K. O'Malley, R. Bangera, Shou de Lin, D. Reeves, and W.E. Walsh. Designing the market game for a trading agent competition. *IEEE Internet Computing*, 5(2):43–51, 2001.

[Vic61] W. Vickrey. Counterspeculations, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[VV03] S. De Vries and R. Vohra. Combinatorial auctions: A survey. *Journal on Computing*, volume 15, No. 3:284–309, 2003.

[WC78] D.A. Wismer and R. Chattergy. *Introduction to linear optymization - A Problem Solving Approach*. Elsevier Noth-Holland, 1978.

[Wie] Jan Wielemaker. *The SWI-Prolog web site*. Documentation and system available via WWW from `http://www.swi-prolog.org/`, (strona dostępna w dniu, 1.05.2008).

[WWWMM01] M. Wellman, W. Walsh, P. Wurman, and J. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[XSW05] M. Xia, J. Stallaert, and A. B. Whinston. Solving the combinatorial double auction problem. *European Journal of Operational Research*, 164(1):239–251, 2005.