Radosław Pytlak
Politechnika Warszawska, Instytut Automatyki i Robotyki

Wojciech Stecz
Wojskowa Akademia Techniczna, Wydział Cybernetyki

Modele dla potrzeb rozwiązywania zadań optymalizacji taryf

Models for solving the tariff optimization problem

W artykule zaprezentowano metody optymalizacji stawek taryfowych telefonii komórkowej z punktu widzenia klienta. Klient stara się minimalizować swoje miesięczne obciążenia rachunkami poprzez dobór optymalnej taryfy telekomunikacyjnej. Zaprezentowano model przydatny dla klientów firm telekomunikacyjnych, które są większymi spółkami i dla których konstrukcje taryf są bardziej złożone. Opisano przykładowe modele całkowitoliczbowe (MIP) i ich rozszerzenia na programowanie z ograniczeniami (CLP). Podano przykłady w językach ECLiPSe i ILOG CLP.

Słowa kluczowe:
optymalizacja taryf, programowanie z ograniczeniami, minimalizacja kosztu.

We present the methods of telecommunication tariff optimization from a point of client's view. A client which wants to minimize his monthly fees tries to choose a proper tariff model. In case of large companies these models are more complicated and the optimization models should be used. We describe a simple MIP models and their modifications solved with CLP solvers. All the examples were solved with ILOG and ECLiPSe MIP and CLP solvers.

Key words:
tariff optimization, constraint logic programming, cost minimization.

Wprowadzenie

Problem poruszany w pracy dotyczy wyboru przez klienta sieci komórkowej optymalnej taryfy spośród zbioru ofert przedstawianych przez operatorów telefonicznych. Klient stara się zwyczajowo wybrać taryfę (ofertę) najlepszą, co zwykle oznacza wybór taryfy najtańszej. Struktura planów taryfowych bywa jednak dość złożona, co utrudnia wybór taryfy optymalnej. Plany taryfowe stanowią bowiem kombinację tzw. kontraktów i usług dostępnych w ramach tych kontraktów, spośród których klient musi wybrać konfigurację dla siebie odpowiednią. Przykładowo, w zbiorze usług występują darmowe minuty na połączenia krajowe w ramach sieci, w ramach sieci partnerskich, darmowe SMS i MMS itp. Generalnie liczba usług może być spora, w szczególności, gdy mowa o planach taryfowych oferowanych przedsiębiorstwom. Zwyczajowo zbiory usług sprzedawane są w pakietach zwanych zwykle kontraktami — kontrakt to propozycja cenowa płatna miesięcznie, w ramach której klient pozyskuje możliwość dostępu do określonych usług na ustalonych zasadach. Każde przekroczenia limitów są oczywiście dodatkowo płatne. Problem optymalizacji taryf dla klienta firmy telekomunikacyjnej można więc opisać jako problem wyboru najtańszej konfiguracji kontraktów tak, aby wykorzystać możliwie dużo usług w ramach kontraktów zakupionych bez konieczności dopłat za przekroczenia.

Oczywiście problem optymalizacji stawki taryfowej może być także rozważany z punktu widzenia operatora telekomunikacyjnego, co zresztą zwyczajowo ma miejsce. W takim przypadku operator chce maksymalizować swoje przychody w czasie. Te modele uwzględniają przede wszystkim stałe przychody w określonym czasie, co oznacza, że operator zgodzi się na obniżenie cen usług w ramach kontraktu — w przypadku kontraktów wieloletnich — w celu związania ze sobą klienta. Takie zarządzanie klientem jest dość dobrze opisane w literaturze przedmiotu. Można przyjąć założenie, że operator maksymalizuje swój zysk poprzez wiązanie ze sobą klientów kontraktami wieloletnimi tak, aby obciążyć swoją sieć możliwie maksymalnie (przy założeniu zachowania pewnych buforów rezerwowych). M. Bouhtou i inni opisali różne modele wyceny usług telekomunikacyjnych (Bouhtou, Medori, Minoux,

2011). Alternatywne modele formułują problem optymalizacji taryf jako problem decyzyjnej gry cenowej między operatorami telekomunikacyjnymi. E. Viterbo zaproponował metodę maksymalizacji zysku na podstawie szacowania kosztów *on-line* dla operatora sieci komórkowej (Viterbo, Chiasserini, 2001). M. Bouhtou i inni opisali metodę określania potencjalnych zysków operatora posiadającego określone łącza telekomunikacyjne i pozwalającego klientom z nich korzystać (Bouhtou, Erbs, Minoux, 2007). Metoda oparta jest na założeniu, że podstawą wyceny usługi jest wybór określonych łączy telekomunikacyjnych. Jednak żaden z powyższych modeli nie jest adekwatny dla przypadku, gdy klient firmy telekomunikacyjnej musi sam zdecydować, jaką ofertę powinien wybrać.

W pracy przyjęto założenie, że klient zna swój profil odnośnie do wykorzystania miesięcznego różnych usług telekomunikacyjnych. Aby określić swój profil, klient powinien sprawdzić średniomiesięczne wykorzystanie usług, takich jak rozmowy w ramach sieci, rozmowy między sieciami oraz wykorzystanie usług SMS i MMS.

Należy zauważyć, biorąc pod uwagę oferty na rynku polskim, że stosowane przez operatorów telekomunikacyjnych mechanizmy pakietowania usług i upustów stanowią duży kłopot dla problemu optymalizacji i zwykle można je rozwiązać jedynie metodami programowania w logice z ograniczeniami lub typowymi metodami heurystycznymi, bardzo szeroko wykorzystywanymi w telekomunikacji.

W artykule zaprezentowano różne sformułowania problemu optymalizacji taryf. Zadanie podstawowe liniowe całkowitoliczbowe jest rozszerzane poprzez dodawanie kolejnych ograniczeń do zadania kombinatorycznego, którego nie można rozwiązać metodami programowania matematycznego. Wszystkie zaprezentowane sformułowania zostały zaimplementowane i rozwiązane w środowiskach CISCO ECLiPSe oraz IBM ILOG CPLEX. Środowisko ECLiPSe jest klasycznym środowiskiem CLP. ILOG stanowi środowisko łączące metody programowania matematycznego (MIP/QP) oraz metody optymalizacji z wykorzystaniem propagacji ograniczeń (CP — *Constraint Propagation*).

Sformułowanie problemu

W tej części artykułu zostanie sformułowany problem optymalizacji stawek taryfowych. Na początku pokazane zostanie sformułowanie zadania liniowego całkowitoliczbowego MILP (ang. *Mixed Integer Linear Problem*). Model został zaimplementowany, zadanie zostało rozwiązane w narzędziach do optymalizacji ILOG MIP i ECLiPSe Eplex. W drugim kroku zadanie rozszerzono poprzez wprowadzenie ograniczeń logicznych, a następnie tzw. predykatów globalnych, które można traktować jako złożone ograniczenia podane w postaci procedur przetwarzania. Przykładem takiego ograniczenia jest predykat globalny *alldifferent*. Problem podstawowy całkowitoliczbowy został zamieniony na problem kombinatoryczny rozwiązywany za pomocą technik CLP lub technik hybrydowych łączących zadania MIP i CLP.

Niech będzie dana zmienna y_i oznaczająca liczbę kontraktów typu i-tego oraz zmienna x_{ij} oznaczająca liczbę jednostek usługi j-tej w ramach kontraktu i-tego, które są dostępne w cenie kontraktu.

Problem optymalizacji taryf oznaczany dalej jako **P** może być sformułowany w sposób pokazany poniżej:

$$\min_{x,y} \sum_{i=1}^{m} \left[c_i y_i + c_i^0 v_i + \sum_{j=1}^{n} c_{ij} \max\left[0, x_{ij} - y_i b_{ij}\right] \right] \quad (1)$$

przy ograniczeniach:

$$\sum_{i=1}^{m} x_{ij} = x_j^h, \quad j = 1,...,n \quad (2)$$

$$\sum_{j=1}^{n} x_{ij} \le M^1 y_i, \quad i = 1,..., m \quad (3)$$

$$y_i \le L_i, \quad i = 1,..., m \quad (4)$$

$$y_i \le M^2 v_i, \quad i = 1,..., m \quad (5)$$

$$0 \le x_{ij}, \quad i = 1,..., m, \quad j = 1,..., n \quad (6)$$

Dla $x = \{x_{ij}\}_{i=1,...,m,\, j=1,...,n}$, $y = \{y_i\}_{i=1,...,m}$, c_i jest miesięcznym kosztem kontraktu i-tego, c_{ij} jest kosztem usługi j-tej w ramach kontraktu i-tego, a b_{ij} stanowi darmowy limit usługi j-tej w ramach kontraktu i-tego, który został zakupiony przez klienta i za który klient płaci stałą opłatę miesięczną.

Dodatkowo x_j^h oznacza średnie historyczne wykorzystanie usługi typu j-tego w zadanym czasie (zwykle jest to ostatni rok lub dwa lata — jeżeli takie dane można pozyskać w firmie klienta). c_i^0 określa koszt miesięczny wykorzystania kontraktów z punktu widzenia klienta. Można to traktować jako koszty dodatkowe poza opłatami dla firmy telekomunikacyjnej, jakie ponosi się po nabyciu zestawu kontraktów u danego operatora telekomunikacyjnego. Zmienna v_i opisuje sytuację, gdy klient ponosi dodatkowe koszty po podpisaniu kontraktu (v_i jest zmienną binarną).

Aby opisać problem w sposób kompletny, należy wskazać, że zmienne y są całkowitoliczbowe, a zmienne x są liczbami rzeczywistymi (chociaż nie musi to być regułą). Problem **P** przybiera wówczas postać zadania liniowego całkowitoliczbowego mieszanego MILP. Wprowadzamy także zmienną pomocniczą z_{ij}, która upraszcza nam funkcję celu (pozbywamy się elementu nieróżniczkowalnego; IBM ILOG, 2009):

$$\min_{x,y} \sum_{i=1}^{m} \left[c_i y_i + c_i^0 v_i + \sum c_{ij} z_{ij} \right] \tag{7}$$

przy ograniczeniach:

$$x_{ij} - y_i b_{ij} - z_{ij} \le 0, \quad i = 1,...,m, \quad j = 1,...,n \tag{8}$$

$$0 \le z_{ij}, \quad i = 1,...,m, \quad j = 1,...,n \tag{9}$$

i ograniczeniach (2)–(6).

Rozwiązanie problemu optymalizacji taryf

Modele MIP

W tej części zaprezentowane zostaną różne modyfikacje zadania optymalizacji taryf. Wprowadzone zostaną dodatkowe ograniczenia logiczne oraz predykaty globalne, takie jak *alldifferent* (CISCO Systems, 2006). Ograniczenia logiczne nie są przez nas zamieniane do postaci ograniczeń algebraicznych w celu pokazania możliwości solwerów hybrydowych, chociaż narzędzia takie, jak ILOG mają mechanizmy transformujące ograniczenia logiczne na ograniczenia algebraiczne, co pozwala na korzystanie ze standardowych metod programowania matematycznego przy rozwiązywaniu zadania optymalizacji stawek taryfowych. W ogólnym przypadku ręczne transformowanie ograniczeń logicznych na algebraiczne stanowi dość złożone zagadnienie (Wallace, 2005; Fruwirth, Abdennadher, 2003).

Pokazany podstawowy problem optymalizacji taryf (1)–(6) jest problemem liniowym całkowitoliczbowym, który oczywiście pojawia się w praktyce, ale raczej dla klientów będących osobami fizycznymi. Plany taryfowe dla firm są znacznie bardziej skomplikowane. Oczywiście modele liniowe dotyczą zarówno klientów indywidualnych, jak i firm, systemy bilingowe nie mogą bowiem być nadmiernie obciążane obliczeniami. Ale faktury końcowe generowane są z poziomu hurtowni danych firm telekomunikacyjnych, a tu są możliwe różne systemy upustów i promocji. Problemy optymalizacji sformułowane w postaci problemów liniowych są najbardziej pożądane ze względu na maksymalne wyniki, jakie się uzyskuje.

Sformułowanie problemu w języku ILOG CPLEX

W tym punkcie przedstawiono sformułowanie problemu optymalizacyjnego, który zawiera jedynie ograniczenia algebraiczne. Problem zamodelowano w języku OPL narzędzia ILOG — nowoczesnego środowiska do modelowania i rozwiązywania zadań programowania całkowitoliczbowego oraz zadań kombinatorycznych.

Poniżej pokazano sposób definiowania parametrów i zmiennych w narzędziu ILOG.

```
int NbContracts =...;
int NbServices =...;

range Contracts = 1.. NbContracts;
range Services = 1.. NbServices;

float ContractCosts[Contracts] =...;
float ContractServicesCosts[Contracts, Services]
=...;
float ContractInitialCosts[Contracts] =...;
float Xh[Services] =...;
float M[Contracts] =...;
float b[Contracts, Services] =...;

dvar int+ y[Contracts];
dvar int+ z[Contracts, Services];
dvar int+ x[Contracts, Services];
dvar int+ v[Contracts] in 0.. 1;
```

Zmienne definiowane jako dvar int+ oznaczają zmienne o dziedzinach będących liczbami całkowitymi dodatnimi.

Funkcja celu (7) może być zamodelowana w postaci:

```
minimize
  sum (i in Contracts) ContractCosts[i]*y[i] +
  sum (i in Contracts, j in Services)
    ContractServicesCosts[i, j]*z[i, j] +
  sum (i in Contracts) ContractInitialCosts[i]*v[i];
```

Ograniczenia (2)–(6) oraz (8)–(9) zamodelowano jako:

```
subject to {
  forall (j in Services)
    c_hist_data:
      sum (i in Contracts) x[i][j] == Xh[j];
  forall (i in Contracts)
    c_yx_connection:
```

```
    sum (j in Services) x[i][j] <= M1*y[i];
  forall (i in Contracts)
    c_contracts_limit: y[i] <= M[i];
  forall (i in Contracts)
    sum (j in Services) z[i][j] >= 0.0;
  forall (i in Contracts, k in Services)
    x[i][k] – y[i]*b[i][k] – z[i][k] <= 0.0;
  forall (i in Contracts)
    c_v_data: y[i] <= M2*v[i];
}
```

W języku OPL możemy opisywać zarówno zadania całkowitoliczbowe (MIP, MILP, MIQP), jak i zadania rozwiązywane przez propagację ograniczeń narzędziami CLP. Należy podkreślić to, że opracowano interfejsy pomiędzy językiem OPL i popularnymi językami programowania, jak: NET, JAVA czy C++.

Sformułowanie problemu w ECLiPSe CLP

Programowanie z ograniczeniami (CLP) stanowi metodę alternatywną, którą możemy wykorzystać w przypadku wyznaczania optymalnych planów taryfowych. CLP wykorzystywane jest głównie w rozwiązywaniu problemów kombinatorycznych, gdzie zdefiniowano ograniczenia inne niż algebraiczne — mogą to być ograniczenia nawet w postaci proceduralnej. Najczęściej spotykane ograniczenia w takiej postaci to tzw. predykaty globalne o nazwach *alldifferent*, *atleast*, *atmost*, które opisano szeroko w literaturze. Przykładowo predykat *atleast* nakłada ograniczenie, aby co najmniej *n* elementów wektora podanego jako parametr wejściowy predykatu miało na wyjściu wartość określoną *a-priori*.

Rachunek CLP stanowi uogólnienie rachunku programowania w logice (LP). Jedyną modyfikacją jest rozszerzenie o zbiory ograniczeń, które są dodawane do rachunku, proces zwany unifikacją jest więc zastąpiony przez rozwiązywanie ograniczeń. Ograniczenia pojawiają się w przesłankach i konkluzjach reguł. Zdania LP i CLP są definiowane w identyczny sposób. Syntaktyka rachunków jest podana w opracowaniu (Fruwirth, Abdennadher, 2003).

Problem optymalizacyjny **P** został zamodelowany w ECLiPSe w postaci przygotowanej do wykorzystania solwera CLP. Zmienne i parametry definiowane są niemal identycznie jak w przypadku ILOG (IBM ILOG, 2009). Definicje ograniczeń i funkcji celu pokazane są poniżej.

Język wykorzystywany do modelowania w ECLiPSe stanowi uogólnienie i rozszerzenie języka Prolog, który nie pozwala na definiowanie ograniczeń (inne wady Prologu podano szeroko w literaturze). Warto zwrócić uwagę, że w przypadku wyrażeń arytmetycznych w ECLiPSe należy stosować funkcję/predykat *eval ()* w celu wskazania solwerowi konieczności uziemienia zmiennych (solwer w sytuacji, gdy nie może tego wykonać i nie udaje mu się przypisać wartości do wszystkich zmiennych wyrażenia, odkłada wyrażenie na stos i kontynuuje poszukiwania, natomiast Prolog w takim przypadku zgłasza błąd; CISCO Systems, 2006).

Proces poszukiwania rozwiązania dla problemów optymalizacji taryf opisany jest w literaturze (Bouhtou, Erbs, Minoux, 2007; Bouhtou, Medori, Minoux, 2011). Techniki modelowania pokazano między innymi w: Nemhauser, Wolsey, 1988 i Pytlak, Stecz, 2007.

```
dim (Y, [NumContracts]),
Y:: 0.. Nc,
dim (V, [NumContracts]),
V:: 0.. 1,
dim (X, [NumContracts, NumServices]),
X:: 0.. Ns,
dim (Z, [NumContracts, NumServices]),
Z:: 0.. Ns,

/* ograniczenia */
% constraint no.
(for(I, 1, NumContracts), param(NumServices,
X, Y, Z, B) do
 (for (J, 1, NumServices), param (X, Z, Y, B, I) do
   eval (X[I, J])-eval(Y[I])*B[I, J]-eval(Z[I, J])
   #=< 0
)),
% constraint no.
(for(I, 1, NumContracts), param(Y, M) do eval
(Y[I])#=< M[I]),
% constraint no.
(for(J, 1, NumServices), param(NumContracts,
Xh, X) do
  Xj is X[1.. NumContracts, J],
  eval(sum(Xj))#= Xh[J]),
% constraint no.
(for (I, 1, NumContracts), param (NumServices, Y, X, M1) do
  Xi is X[I, 1.. NumServices],
  eval (sum (Xi)) #=< M1*Y[I]),
% coinstraint no.
(for(I, 1,NumContracts), param(NumServices,
Z) do
  Zi is Z[I, 1.. NumServices],
  eval(sum(Zi))#>= 0),
% coinstraint no.
(for(I, 1,NumContracts), param(Y, V, M2)
do
  eval(Y[I])#=< M2*eval(V[I])),
```

```
/* kryteria optymalizacji */
(for(I, 1, NumContracts),
fromto(0, In1, In1+ContractCosts[I] * Y[I],
Costs1),
param(Y, ContractCosts) do true),

(for(I, 1, NumContracts) * for(J, 1, NumServi-
ces),
fromto (0, In, Out, Costs2),
param(ServiceCosts, Z) do
  Out = In + ServiceCosts[I, J] * Z[I, J]),

(for (I, 1, NumContracts),
fromto (0, In1, In1+ContractInitialCosts[I] *
V[I], Costs3),
param (V, ContractInitialCosts) do true),

TotalCosts #=eval(Costs1)+eval(Costs2)
+eval(Costs3),
term_variables ([V, Y, X, Z, TotalCosts],
Vars),


/* algorytm poszukiwania rozwiązania */
bb_min(search(Vars, 0, first_fail, indoma-
in_min, bbs (100),
[backtrack (Backtracks)]), TotalCosts,
bb_options{strategy: dichotomic, timeout:
260})
```

Funkcja *bb_min* inicjalizuje działanie solwera B&B dla zmiennych całkowitoliczbowych i rzeczywistych typu IC (biblioteka ECLiPSe wykorzystywana w algorytmach propagacji ograniczeń, co pozwala na łączenie obu technik — poszukiwań metodami znanymi z optymalizacji liniowej i poszukiwań opartych na metodachy propagacji ograniczeń). Wykorzystanie biblioteki IC pozwala na wygodną komunikację solwerów w ramach środowiska hybrydowego. W przypadku zadań CP metodą inicjalizującą solwer CLP jest metoda *search*, która w omawianym przypadku była konfigurowana tak, aby przeglądać w pierwszej kolejności zmienne o najmniejszych dziedzinach.

Dodatkowe ograniczenia: logiczne i predykaty globalne

W wielu problemach biznesowych trudno jest sformułować liniowe zadanie optymalizacji lub zadanie, które zawiera jedynie ograniczenia algebraiczne. Czasami zdarza się, że ograniczenia przybierają postać dodatkowych procedur obliczeniowych. Przykładem jest predykat globalny *alldifferent*, który jest omówiony w dalszej części pracy.

Przykładowe ograniczenia logiczne wykorzystywane w pracy podano poniżej.

$$y_j \geq \alpha \rightarrow y_k \leq \beta \tag{10}$$

$$x_j^h \geq \alpha \rightarrow y_k \leq \beta \tag{11}$$

Oba ograniczenia (10, 11) uzależniają wartości zmiennej y_k od wartości innych zmiennych w zadaniu. Zarówno ILOG CPLEX, jak i ECLiPSe wykorzystują zmienne logiczne w swoich solwerach CLP. ILOG CPLEX potrafi dodatkowo automatycznie zamienić ograniczenia logiczne na algebraiczne i przekształcić zadanie hybrydowe w zwykłe zadanie całkowitoliczbowe. W omawianym przypadku parametry α, β są związane z danymi historycznymi dotyczącymi wykorzystania usług przez klienta.

Predykat globalny *alldifferent* wymusza, aby każdy z elementów wektora podanego na wejściu procedury miał na wyjściu inną wartość. Predykat *alldifferent* pojawia się w wielu praktycznych zadaniach kombinatorycznych i metody jego implementacji są bardzo dokładnie badane i opisane. Klasycznym przykładem, gdzie można go wykorzystać, jest problem n królowych, które nie mogą się wzajemnie szachować na szachownicy o rozmiarach $n \times n$. Jednym ze sposobów efektywnego zamodelowania problemu jest zastosowanie właśnie predykatu globalnego *alldifferent*.

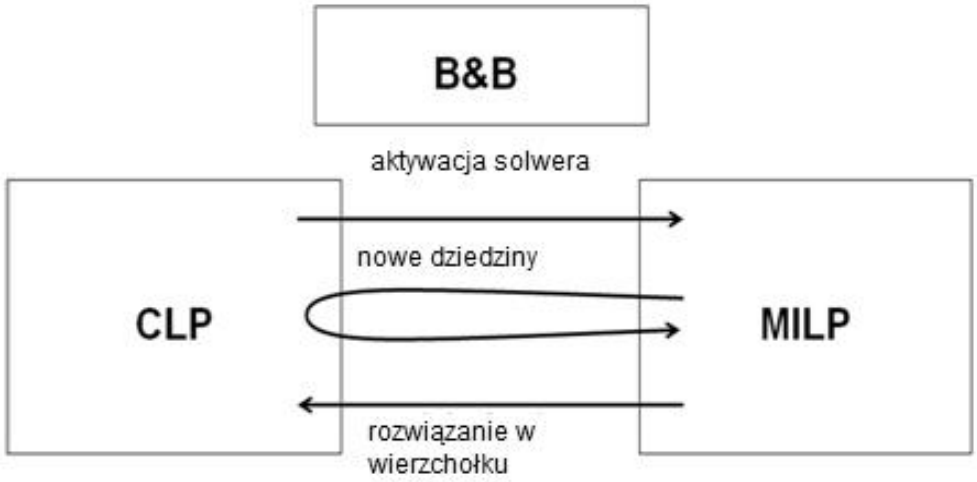
Sformułowanie problemu na solwer hybrydowy (MIP i CLP)

Solwery hybrydowe łączące w sobie techniki optymalizacji całkowitoliczbowej, jak i techniki oparte na propagacji ograniczeń muszą komunikować się w efektywny sposób w celu szybkiego znalezienia rozwiązania dopuszczalnego, a w dalszej kolejności rozwiązania optymalnego/suboptymalnego. Aby usprawnić komunikację między solwerami, należy połączyć techniki przeszukiwania drzewa podziału i ograniczeń z technikami CP poprzez umiejętne sterowanie wywołaniami tych drugich na skutek zdarzeń systemowych. W tym przypadku otrzymujemy narzędzia uwzględniające wszystkie ograniczenia zadania bez względu na ich postać. Solwer hybrydowy musi potrafić wykrywać niespójności ograniczeń i wskazywać sytuacje, gdy nie ma rozwiązań dopuszczalnych.

W przypadku takiego narzędzia, jak ECLiPSe CLP instancja solwera MIP jest automatycznie powoływana w momencie rozpoczęcia obliczeń. Narzędzie wykorzystuje metodę Branch&Bound jako główny mechanizm sterujący optymalizacją i zajmuje się przekazywaniem sterowania do solwera CLP po uzyskaniu rozwiązania w wierzchołku drzewa B&B dla zadania całkowitoliczbowego. Solwer CLP wnosi do układu dodatkowe ograniczenia, które zawężają przestrzeń rozwiązań. Jego zadaniem jest zasadniczo ograniczenie przestrzeni rozwiązań w początkowej fazie rozwiązania zadania, co nie zawsze musi się udać zrealizować. W literaturze definiuje się zbiór zdarzeń, po jakich mechanizm sterujący powinien przekazać sterowanie do solwera CLP i odwrotnie. Dotyczy to przede wszystkim sytuacji, gdy po wywołaniu CLP zawężenie dziedzin spowodowane propagacją ograniczeń eliminuje część elementów wektora rozwiązania w danym wierzchołku (Wallace, Schimpf, 2002).

Schemat komunikacji między solwerami pokazano na rysunku 1.

Rysunek 1
Schemat działania solwera hybrydowego MIP/CLP



Źródło: opracowanie własne.

Podstawowe kroki konfiguracji i obliczeń solwera hybrydowego to:

- ustalenie ograniczeń,
- określenie metody przeszukiwania przestrzeni rozwiązań,
- propagacja ograniczeń — zależna od dziedzin zmiennych: IC (liczby rzeczywiste i całkowite), FD (zbiór elementów skończonych),
- rozwiązanie problemu zrelaksowanego MIP,
- sprawdzenie rozwiązania w węźle, określenie granicy górnej i dolnej dla możliwego rozwiązania zadania.

Kroki 3–4 powtarzają się cyklicznie w trakcie obliczeń.

Przykładowy problem optymalizacyjny **P** podano poniżej. **P** opisano jako zadanie z ograniczeniami algebraicznymi, logicznymi i ograniczeniami w postaci procedur (w CP określa się takie ograniczenia jako predykaty globalne).

```
dim (Y, [NumContracts]),
[eplex, ic]: (Y[NumContracts] $:: 0.0.. T),
[eplex, ic]: (integers(Y[1.. NumContracts])),
dim(V,[NumContracts]),
[eplex]: (V[NumContracts] $:: 0.0.. 1.0),
[eplex]: (integers (V[1.. NumContracts])),
dim(X,[NumContracts, NumServices]),
[eplex]:(X[NumContracts,NumServices]
$:: 0.0.. M),
dim(Z,[NumContracts, NumServices]),
[eplex]:(Z[NumContracts,NumServices]
$:: 0.0.. N),
```

Solwerem MIP w systemie ECLiPSe jest Eplex. Eplex MILP może być podmieniony na solwery komercyjne, w tym CPLEX, solwery z projektu COIN-OR. W naszym przypadku używano także solwera całkowitoliczbowego z projektu COIN-OR o nazwie COIN-OR CBC.

Jest rzeczą niezwykle ważną, aby przyporządkować ograniczenia do każdego z typów solwerów. Błędne przypisanie ograniczeń spowoduje dłuższe działanie narzędzia lub uniemożliwi pozyskanie rozwiązania.

```
/* MIP constraints */
(for (I, 1, NumContracts), param (NumServices, X, Y, Z, B) do
  (for(J, 1, NumServices), param(X, Z, Y, B, I) do
   eplex: (X[I, J] - Y[I]*B[I, J] - Z[I, J] $=< 0.0))),

(for(I, 1, NumContracts), param(NumServices, X, Z) do
 (for(J, 1, NumServices), param(X, Z, I) do
   eplex: (X[I, J] $>= 0.0),
   eplex: (Z[I, J] $>= 0.0))),

(for(I, 1, NumContracts), param(Y, V) do
[eplex, ic]: (Y[I] $>= 0.0),
[eplex, ic]: (Y[I] $=< 100.0),
eplex: (V[I] $>= 0.0),
eplex: (V[I] $=< 1.0)),

(for(J, 1, NumServices), param(NumContracts, Xh, X) do
 eplex: (sum (X[1.. NumContracts, J]) $= Xh[J])),

(for(I, 1, NumContracts), param(NumServices, Y, X, M1) do
 eplex: (sum (X[I, 1.. NumServices]) $=< M1*Y[I])),

(for(I, 1, NumContracts), param(Y, V, M2) do
 eplex: (Y[I] $=< M2*V[I])),

/* CLP constraints */
ic: (Y[9] $=< 10),
ic: (alldifferent (Y[1.. 3])),
ic: (Cost $>= 0.0),
/* optimization criteria */
!!! The same as in MIP solver case

eplex: (TotalCosts $= Costs1 + Costs2 + Costs3),

/* search algorithm */
eplex_solver_setup(
 min (TotalCosts), Cost, [sync_bounds(yes)], inst]),

bb_min((labeling (Y), eplex_get (cost, Cost)), Cost, bb_options{strategy: continue})
```

Mechanizm sterujący B&B rozwiązuje przede wszystkim zadanie całkowitoliczbowe, które stanowi bazę dla dalszego poszukiwania rozwiązania. W dostępnej literaturze nie znaleziono alternatywnych podejść. Wiąże się to z efektywnością metod poszukiwania rozwiązań dla problemów liniowych i liniowych całkowitoliczbowych. Po uzyskaniu rozwiązania w wierzchołku następuje proces określany w języku angielskim jako *labeling* (co oznacza określanie wartości elementów wektora rozwiązań, które może modyfikować solwer CLP). Określanie rozwiązania przez CLP polega na propagacji ograniczeń, która jest bogato opisana w literaturze. Należy pamiętać o tym, że gdy na skutek propagacji ograniczeń zmienia się wartość elementu w wektorze rozwiązania Y, to mechanizm B&B restartuje solwer MIP z wektorem Y w trybie gorącego restartu. Zapewnia to efektywny proces poszukiwania rozwiązania w otoczeniu rozwiązania dotychczas znalezionego. Proces jest kontynuowany iteracyjnie do momentu znalezienia rozwiązania, którego nie można poprawić, lub określenia sytuacji, w której rozwiązanie nie występuje.

Wyniki numeryczne

Zaprezentowany model został zaimplementowany w dwóch narzędziach IBM ILOG oraz ECLiPSe CLP. W tabelach 1–4 przedstawione zostały wyniki dla problemów małej i średniej skali.

Relatywnie dobre wyniki dla problemu drugiego można wytłumaczyć faktem zawężenia przestrzeni rozwiązań po wprowadzeniu dodatkowych ograniczeń logicznych. To narzuca wniosek, że dodatkowe ograniczenia logiczne przyspieszają działanie systemu. Nie jest to jednak prawdą w ogólnym przypadku.

W tabeli 2 zaprezentowano wyniki optymalizacji dla problemu rozwiązywanego poprzez techniki propagacji ograniczeń i solwer CLP. Wszystkie zmienne są liczbami całkowitymi, co poprawia proces przeszukiwania przestrzeni rozwiązań (techniki CP traktują liczby rzeczywiste jako przedziały wartości, co spowalnia przeszukiwanie). W przypadku, gdy zmienne są rzeczywiste, wynik optymalizacji otrzymuje się po znacznie dłuższym czasie oczekiwania.

Wyniki eksperymentów pokazanych w artykule wskazują na bardzo duże znaczenie poprawnego ustawienia parametrów początkowych solwera oraz pierwszego rozwiązania dopuszczalnego (Pytlak, Stecz, 2007).

Pokazane wyniki udowadniają, że stosowanie solwerów CLP w przypadku zadań optymalizacji klasy liniowej lub liniowej całkowitoliczbowej nie jest dobrym rozwiązaniem ze względu na nieporównywalnie słabsze wydajnościowo metody propagacji ograniczeń.

Dlatego też znaczenia w rozwiązywaniu zadań

tego typu nabierają solwery hybrydowe, które umożliwiają rozwiązanie zadań MIP metodami opartymi o algorytmy simpleks lub metody punktu wewnętrznego, a zadania z ograniczeniami innymi niż algebraiczne rozwiązywane są metodami propagacji ograniczeń.

Przykładem stosowania ograniczeń innych niż algebraiczne jest predykat *alldifferent*.

W przypadku, gdy nie nakłada się ograniczeń na liczbę możliwych do wybrania kontraktów, solwer CLP przeszukuje przestrzeń rozwiązań efektywnie. Jeżeli jednak liczba możliwych kontraktów, jakie można wybrać, jest ograniczona (czyli dodano nowe ograniczenia zadania), wówczas czas rozwiązania zadania staje się znacząco dłuższy. W przypadku, gdy istotne jest pozyskanie

Tabela 1
Wyniki dla MIP ILOG CPLEX

Optymalizacja problemów średniej wielkości Liczba zmiennych: 1327 (13 kontraktów i 50 usług)			
MILP problem	Czas CPU[s]	Liczba iteracji	Liczba odcięć
Bez ograniczeń logicznych	0.47	1056	188
Z ograniczeniami logicznymi	0.20	373	11

Źródło: opracowanie własne.

Tabela 2
Wyniki dla CLP ILOG CPLEX

Optymalizacja problemów średniej wielkości Liczba zmiennych: 1327 (13 kontraktów i 50 usług)			
CLP problem	Czas CPU [s]	Liczba gałęzi	Liczba nawrotów
Bez ograniczeń logicznych	120.0	216.000	>97.000
Z ograniczeniami logicznymi	120.0	214.000	>97.000
Z ograniczeniami logicznymi oraz predykatem *alldifferent* (Y[1..3])	(limit czasu przekroczony)	179.000	>70.000

Źródło: opracowanie własne.

Tabela 3
Wyniki dla CLP ECLiPSe

Optymalizacja problemów średniej wielkości Liczba zmiennych: 1620 (10 kontraktów i 80 usług)		
CLP problem	Czas CPU [s]	
	Strategia przeszukiwania: *most_constrained*	Strategia przeszukiwania: *occurrence*
Bez ograniczeń logicznych	23.0	18.85
Z ograniczeniami logicznymi	20.85	(limit czasu przekroczony)
Z ograniczeniami logicznymi oraz predykatem *alldifferent* (Y[1..3])	(limit czasu przekroczony)	(limit czasu przekroczony)

Źródło: opracowanie własne.

Tabela 4
Wyniki dla rozwiązania hybrydowego ECLiPSe

Optymalizacja problemów średniej wielkości Liczba zmiennych: 1620 (10 kontraktów i 80 usług)		
CLP problem	Czas CPU [s]	
	Metoda przeszukiwania B&B: *Continue*	Metoda przeszukiwania B&B: *Dichotomic*
Z ograniczeniami logicznymi	13.0	25.0
Z ograniczeniami logicznymi oraz predykatem *alldifferent* (Y[1..3])	39.9	82.0

Źródło: opracowanie własne.

wyników w ciągu pojedynczych sekund, zadanie takie może nie zostać rozwiązane.

W opisywanym przypadku strategia przeszukiwania przestrzeni rozwiązań oparta jest na przeszukiwaniu typu *most_constrained* (w pierwszej kolejności wyznacza się wartości dla zmiennych o najmniejszych dziedzinach).

Strategia typu *occurrence* polega na rozpoczęciu przeszukiwania od dziedzin, na których opiera się największa liczba ograniczeń.

Inną często stosowaną strategią poszukiwania jest *dbs* (*depth bounded search*). Strategia ta polega na eksploracji pierwszych *k* gałęzi w drzewie poszukiwania rozwiązania w sposób kompletny. Po tym kroku wybierana jest inna metoda przeszukiwania przestrzeni rozwiązań.

Obecność predykatu *alldifferent* powoduje znaczące spowolnienie przeszukiwania przestrzeni rozwiązań. W praktyce bowiem solwer każdorazowo uruchamia dość złożony algorytm wyznaczania różnych wartości wektora (metoda/predykat *alldifferent*). Predykaty są bowiem opisanymi w literaturze najczęściej stosowanymi w optymalizacji ograniczeniami o postaci kombinatorycznej. Stąd ich obecność w zadaniach znacząco spowalnia uzyskanie rozwiązania. Jeżeli więc nie ma absolutnej konieczności stosowania predykatów globalnych, to nie powinno się ich umieszczać w zadaniach optymalizacji.

Wnioski

W artykule zaprezentowano praktyczny problem modelowania taryf klientów sieci telekomunikacyjnych. Opisany problem wyznaczania taryf optymalnych pokazano z punktu widzenia klienta firmy telekomunikacyjnej. Ponieważ firmy telekomunikacyjne oferują klientom bardzo złożone umowy dotyczące taryf telekomunikacyjnych, potencjalny klient powinien dysponować narzędziami do wyznaczania taryf optymalnych z jego punktu widzenia.

Biorąc pod uwagę złożoność procesu modelowania taryfy telekomunikacyjnych, wykorzystaliśmy w pracy techniki programowania z ograniczeniami (CLP). Omówiono także funkcjonalność solwerów do rozwiązania zadań klasy CLP. W szczególności dotyczy to narzędzi IBM ILOG oraz narzędzia ECLiPSe CLP.

Literatura

Bouhtou, M., Erbs, G., Minoux, M. (2007). Joint optimization of pricing and resource allocation in competitive telecommunications networks. *Networks*, (50), 37–49.

Bouhtou M., Medori J., Minoux M. (2011). Mixed Integer Programming model for pricing in telecommunication, INOC 2011. Hambourg: France.

Bouhtou M., Hoesel S., Kraaaij A., Lutton J. (2003). Tariff optimization in networks. *INFORMS Journal on Computing*, *19*, (3), 458.

CISCO Systems, (2006), ECLiPSe User Manual ver. 6.0.

CISCO Systems, (2006), ECLiPSe Tutorial Introduction ver. 6.0.

Fruwirth T., Abdennadher S. (2003). *Essentials of constraint programming*. Springer.

Goncalves, J.P.M. Ladanyi, L. (2005). An Implementation of a Separation for Mixed Integer Rounding Inequalities. *IBM Research Report RC23686* August 2.

IBM ILOG, (2009), ILOG OPL Language User's manual ver. 6.3.

IBM ILOG, (2009), User's manual for CPLEX ver. 12.2.

Kiwiel, K. C. (1985) Methods of descent for nondifferentiable optimization. Lecture Notes in Mathematics Heidelberg: Springer-Verlag.

Nemhauser, G., Wolsey, L. (1988). *Integer and Combinatorial Optimization*. Wiley Interscience.

Pytlak, R., Stecz W. (2007) *Tariff optimization problem*, IFIP 2007. Cracow: EAIiE-AGH.

Simonis, H. (2005). *Developing Applications with ECLiPSe*, IC-Parc Technical Report-03-2.

Wallace, M. (2005). Hybrid algorithms, local search and ECLiPSe. CP Summer School.

Wallace, M., Schimpf, J. (2002). Finding the right hybrid algorithm — A combinatorial meta-problem. *Annals of Mathematics and Artificial Intelligence*, (34), 259–269.

Viterbo, E., Chiasserini, C. (2001). Dynamic Pricing in Wireless Networks, 12th International Symposium on Personal, Indoor and Mobile Radio Communications. USA California: San Diego.