ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2000
Seria: AUTOMATYKA z. 131 Nr kol. 1476

Tomasz SZCZYGIEŁ
Politechnika Śląska

ROZWIĄZYWANIE PROBLEMÓW UPAKOWANIA KĄTOWNIKÓW NA PŁASZCZYŹNIE I W PRZESTRZENI

Streszczenie. W artykule przedstawiono rozwiązywanie zagadnień upakowania figur płaskich i przestrzennych. Rozważane zagadnienia obejmują zarówno figury proste, jak i złożone. Omówiona została również złożoność obliczeniowa przytoczonych przykładów. Rozwiązanie wszystkich omawianych przykładów przedstawione jest za pomocą programowania w logice z ograniczeniami - w CHIP-ie.

SOLVING A TWO-DIMENSIONAL AND THREE-DIMENSIONAL ANGLE PACKING PROBLEM

Summary. The paper is about solving two-dimensional and three-dimensional packing problems for simple and complex figures. For all presented problems the computational complexity has been analyzed. This is followed by their solutions presented in the constraint logic programming language– CHIP.

Wstęp

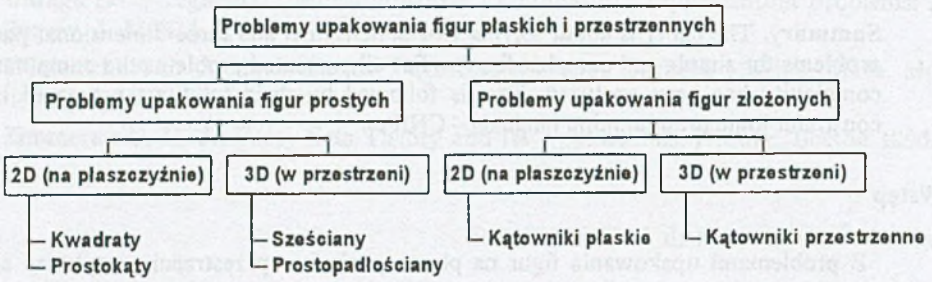
Z problemami upakowania figur na płaszczyźnie i w przestrzeni spotykamy się na co dzień, od stosunkowo banalnego pakowania najpotrzebniejszych rzeczy do walizki przed podróżą, poprzez transport ciężarówkami różnego rodzaju paczek, do załadunku sprzętu i wyposażenia kosmonautów na prom kosmiczny w czasie wyprawy orbitalnej.

We wcześniejszych artykułach konferencyjnych [Constraint Programming for Decision and Control CPDC'99 oraz CPDC'2000] prezentowane były szczegóły sposobów rozwiązywania problemów upakowania figur prostych i złożonych, na płaszczyźnie i w przestrzeni za pomocą programowania w logice z ograniczeniami (Constraint Logic Programming) w CHIP'ie. Wspomniane wyżej artykuły [11 i 12] odnoszą się do problemów załadunku i transportu szynoprzewodów oraz przewodów okapturzonych stosowanych głównie w energetyce do wyprowadzenia mocy z generatorów w elektrowniach. Proste odcinki szynoprzewodów można zamodelować prostokątami na płaszczyźnie i prostopadłościanami w przestrzeni, natomiast krzywe elementy szynoprzewodów można modelować kątownikami na płaszczyźnie i w przestrzeni. W przypadku załadunku i transportu tych elementów dłużycą problem sprowadza się do upakowania kątowników i

prostokątów na płaszczyźnie, natomiast załadunek i transport kontenerami jest przypadkiem upakowania kątowników i prostopadłościanów w przestrzeni. W tym artykule została dokonana klasyfikacja problemów upakowania wg różnych kryteriów, następnie przedstawiono wyniki rozwiązania problemu upakowania pięciu figur prostych i złożonych zarówno na płaszczyźnie, jak i w przestrzeni. W dalszej części omówiona jest złożoność obliczeniowa prezentowanych przykładów. W artykule przedstawiono również trzy podklasy optymalnego rozmieszczenia figur różniące się sformułowaniem problemu upakowania.

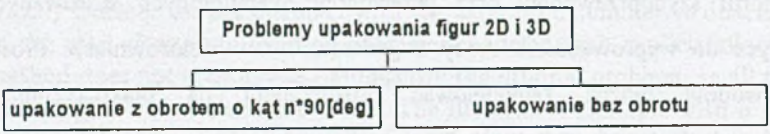
1. Klasyfikacja problemów upakowania

Problemy upakowania figur (lub inaczej optymalnego rozmieszczenia figur) można podzielić wg różnych kryteriów. Jednym z nich jest podział wg rodzaju upakowywanych figur, co przedstawia rys.1.



Rys.1. Klasyfikacja problemów upakowania figur wg ich rodzaju
Fig.1. Taxonomy of basic packing problems

Mówiąc o upakowywaniu figur prostych lub złożonych, należy rozróżniać upakowanie figur płaskich i przestrzennych. Na ostatnim dolnym poziomie rys.1. wymienione są figury, które występują w przykładach omawianych w następnych punktach artykułu. Innym podziałem problemów upakowania jest podział w zależności od sposobu umieszczenia danej figury na płaszczyźnie czy w przestrzeni. Taką klasyfikację przedstawia rys.2.



Rys.2. Klasyfikacja problemów upakowania figur wg sposobu upakowania
Fig.2. Taxonomy of models of packing problems

Problemy upakowania figur można formułować na trzy różne sposoby, jako:

- minimalizację powierzchni lub przestrzeni upakowania przy danej liczbie figur,
- maksymalizację liczby upakowywanych elementów przy danych wymiarach powierzchni lub przestrzeni upakowania,
- rozmieszczenie danej liczby figur na danej powierzchni lub w przestrzeni upakowania, przy założeniu że dane figury mieszczą się na danej powierzchni lub w danej przestrzeni.

2. Problemy upakowania figur na płaszczyźnie

Omówienie problemu upakowania figur na płaszczyźnie rozpoczniemy od figur prostych bez możliwości obrotu.

Przykład 1. Danych jest pięć prostokątów o wymiarach podanych w tabeli 1. Należy tak upakować dane figury, aby zajmowały najmniejszą powierzchnię, jednocześnie nie zachodziły na siebie.

Tabela 1

Dane do przykładu 1

i	1	2	3	4	5
D_{xi}	4	2	10	4	4
D_{yi}	6	6	4	2	4

Wykorzystując standardowo wbudowane w Chip-a predykaty *cumulative* i *diffn* oraz *min_max* i *append* możliwe jest uzyskanie zwięzłego kodu źródłowego rozwiązującego przykład 1:

```
run:-
        data(Data),
        EndX  :: 1..10,
        EndY  :: 1..10,
        min_max((gen_lists(Data,LX,LY,DX,DY,DF),
                diffn(DF),
                cumulative(LX,DX,DY,unused,unused,EndY,EndX,unused)
                ,
                cumulative(LY,DY,DX,unused,unused,EndX,EndY,unused)
                ,
                append(LX,LY,XY),
                labeling(XY)),EndX+EndY).

gen_lists([],[],[],[],[],[]).
gen_lists([DXH,DYH|DT],[LXH|LXT],[LYH|LYT],[DXH|DXT],[DYH|DYT],
[DFH|DFT]):-
        LXH :: 0..9,
        LYH :: 0..9,
        append([LXH,LYH],[DXH,DYH],DFH),
        gen_lists(DT,LXT,LYT,DXT,DYT,DFT).
```

```
labeling([]).
labeling([H|T]):-
        delete(A,[H|T],B,0,first_fail),
        indomain(A),
        labeling(B).
```

Sposób reprezentacji figury prostej poprzez poszczególne zmienne w predykatach zawarty jest w [11]. Powyżej przedstawiony program napisany w CHIP-ie w pełni rozwiązuje problem postawiony w przykładzie 1. Ważnym elementem, na który warto zwrócić uwagę jest predykat *labeling*, który dokonuje ukonkretnienia wartości poszczególnych zmiennych. Od budowy tego predykatu w znacznej mierze zależy szybkość znajdywania rozwiązania. Innym ważnym elementem jest zamknięcie w standardowym predykacie *min_max* predykatów odpowiedzialnych za: generację zmiennych (*gen_lists*), standardowych ograniczeń (*diffn* i *cumulative*) oraz wspomnianego już predykatu (*labeling*). Dzięki takiej konstrukcji programu wynik, który otrzymujemy, jest optimum globalnym,a nie lokalnym.

Dla przedstawionego przykładu program uruchomiony na komputerze Pentium II 300 MHz, 96 MB RAM rozwiązanie znalazł zaledwie po $80*10^{-3}$ [s], co przedstawia rys.3.

Następny przykład wprowadza „stopień swobody" w postaci możliwości obrotu upakowywanej figury o kąt będący wielokrotnością kąta 90°.

Przykład 2: Danych jest pięć prostokątów o wymiarach podanych w Tabeli 2. Należy tak upakować dane figury z możliwością obrotu, aby zajmowały najmniejszą powierzchnię, jednocześnie nie zachodziły na siebie.

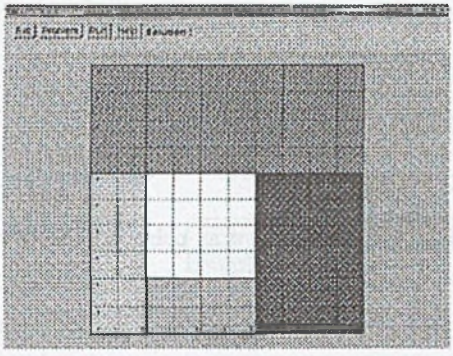
Tabela 2

Dane do przykładu 2

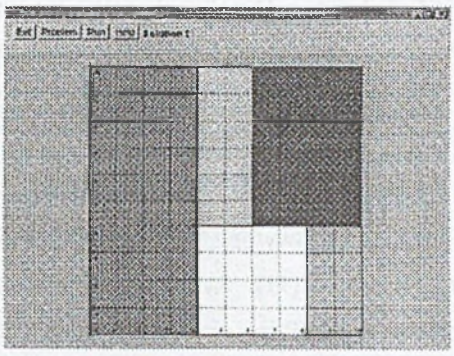
i	1	2	3	4	5
Długości boków	4 * 6	2 * 6	10 * 4	4 * 2	4 * 4

Program rozwiązujący tego typu problemy jak w przykładzie 2, znajduje się w [11]. Warto tutaj jednak dodać, że wprowadzając stopień swobody w postaci możliwości obrotu o kąt będący wielokrotnością kąta 90° kod źródłowy powiększył się jedynie o dwie linijki. Dla omawianego przykładu program uruchomiony na tym samym komputerze co poprzednio rozwiązanie znalazł po $110*10^{-3}$ [s], rozwiązanie graficzne przedstawia rys.4.

Porównując rys.3 i rys.4. widzimy, że wymiary powierzchni, na której zostały upakowane proste figury płaskie, są identyczne, natomiast różnią się ułożeniem figur wewnątrz powierzchni upakowania.



Rys.3. Rozwiązanie przykładu 1
Fig.3. Solution for Example 1



Rys.4. Rozwiązanie przykładu 2
Fig.4. Solution for Example 2

Kolejny przykład przedstawia figury złożone bez możliwości obrotu.

Przykład 3. Danych jest pięć kątowników o wymiarach podanych w tabeli 3. Należy tak upakować dane figury, aby zajmowały najmniejszą powierzchnię, jednocześnie nie zachodziły na siebie.

Tabela 3

Dane do przykładu 3

Nr kątownika	**1**	**2**	**3**	**4**	**5**
Lista długości kątowników	*[3,8,4,2]*	*[4,1,2,3]*	*[2,5,4,3]*	*[3,2,5,4]*	*[3,10,5,2]*

Szczegóły dotyczące sposobu opisu kątownika przez listę długości boków, ideę rozwiązywania tego typu problemów i kod źródłowy programu znajdują się w [11]. Problem upakowania figur złożonych reprezentowanych tutaj przez kątowniki nie jest rozważaniem czysto teoretycznym, bo w rzeczywistości opisuje on transport dłużycą (ułożenie elementów na płaszczyźnie) szynoprzewodów stosowanych w elektrowniach. Program rozwiązujący ten problem znajduje rozwiązanie optymalne po $130*10^{-3}$ [s] przedstawione na rys.5.

Następny przykład zamyka omówienie problemów upakowania figur na płaszczyźnie, a przedstawia on problem upakowania figur złożonych z możliwością obrotu o kąt n*90°.

Przykład 4. Danych jest pięć kątowników o wymiarach podanych w tabeli 4. Należy tak upakować dane kątowniki z możliwością obrotu o kąt będący wielokrotnością kąta 90°, aby zajmowały najmniejszą powierzchnię, jednocześnie nie zachodziły na siebie.

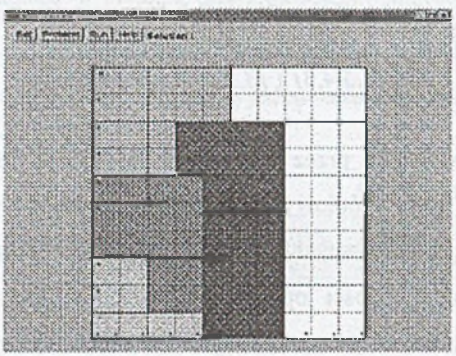
Tabela 4

Dane do przykładu 4

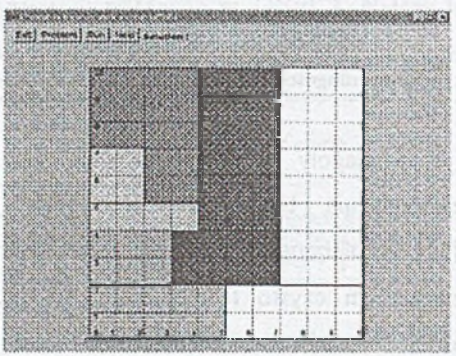
Nr kątownika	6	7	8	9	10
Lista długości kątowników	*[3,8,4,2]*	*[2,3,4,1]*	*[2,5,4,3]*	*[3,4,5,2]*	*[3,10,5,2]*

Szczegółowy opis metody rozwiązania tego typu problemu oraz pełny kod źródłowy znajduje się w [11]. Rys.6 przedstawia rozwiązanie problemu z przykładu 6. Program napisany w CHIP-ie i uruchomiony na komputerze Pentium II 300 MHz, 96 MB RAM znajduje rozwiązanie po 11,130 [s].

Porównując rys.5 i rys.6 widzimy, że wymiary powierzchni upakowania są w obydwu przypadkach identyczne. Jest to prawidłowe zjawisko, ponieważ obydwa przykłady bazują na tym samym zestawie danych, tzn. ta sama liczba kątowników o tych samych wymiarach. Identyczne dane są zabiegiem mającym na celu sprawdzenie poprawności otrzymanego rozwiązania, którym jest minimum globalne powierzchni upakowania. Ułożenie kątowników wewnątrz powierzchni upakowania jest różne, co świadczy o kilku rozwiązaniach będących optymalnym upakowaniem kątowników na płaszczyźnie.



Rys.5. Rozwiązanie przykładu 3
Fig.5. Solution for Example 3



Rys.6. Rozwiązanie przykładu 4
Fig.6. Solution for Example 4

3. Problemy upakowania figur w przestrzeni

Problemy upakowania figur w przestrzeni, podobnie jak w przypadku płaszczyzny rozpoczniemy od najprostszego przypadku, tzn. od upakowania figur prostych bez możliwości obrotu.

Przykład 5. Danych jest pięć prostopadłościanów o wymiarach podanych w tabeli 5. Należy tak upakować dane figury, aby zajmowały najmniejszą przestrzeń, jednocześnie nie przenikały się.

Tabela 5

Dane do przykładu 5

i	1	2	3	4	5
D_{xi}	1	3	2	3	3
D_{yi}	2	2	2	1	1
D_{zi}	2	1	2	1	2

Adekwatny program napisany w CHIP-ie rozwiązujący problem zawarty w przykładzie 5 przedstawiony jest poniżej:

```
run:-
      data(Data),
      EndX :: 1..10,
      EndY :: 1..10,
      EndZ :: 1..10,
      HighX:: 1..10,
      HighY:: 1..10,
      HighZ:: 1..10,
      EndX #<=3,
      EndY #<=3,
      EndZ #<=3,
      min_max((gen_lists(Data,DX,DY,DZ,LX,LY,LZ,DF),
            diffn(DF),
            append(LX,LY,XY),
            append(XY,LZ,XYZ),
            cumulative(LX,DX,DY,unused,unused,HighX,EndX,unused),
            cumulative(LY,DY,DZ,unused,unused,HighY,EndY,unused),
            cumulative(LZ,DZ,DX,unused,unused,HighZ,EndZ,unused),
            labeling(XYZ)),EndX+EndY+EndZ).
```

Wytłuszczone linijki programu są ograniczeniami wymiarów przestrzeni, w której upakowujemy prostopadłościany. Ograniczenia wymiarów przestrzeni znajdują się przed predykatem *min_max*. Taka konstrukcja programu zwiększa szybkość działania, szczególnie gdy mamy do czynienia z dużą liczbą upakowywanych elementów.

Porównując kody źródłowe programów z przykładów 1 i 5 widzimy, że programowanie w logice z ograniczeniami jest wydajnym narzędziem umożliwiającym szybkie rozwiązanie tego typu zagadnień. Komputer z procesorem Pentium II 300 MHz, 96 MB RAM znajduje rozwiązanie przykładu 5 po $180*10^{-3}$ [s], interpretację graficzną przedstawia rys.7.

Kolejny przykład omawia problem figur prostych w przestrzeni z możliwością obrotu o kąty będące wielokrotnością kąta 90° we wszystkich trzech osiach.

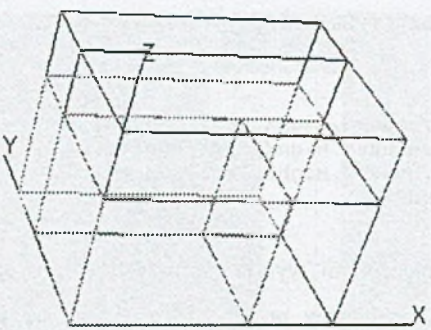
Przykład 6. Danych jest pięć prostopadłościanów o wymiarach podanych w tabeli 6. Należy upakować dane figury z możliwością obrotu o wielokrotność kąta 90° we wszystkich trzech osiach, tak aby zajmowały najmniejszą przestrzeń, jednocześnie nie przenikając się.

Tabela 6

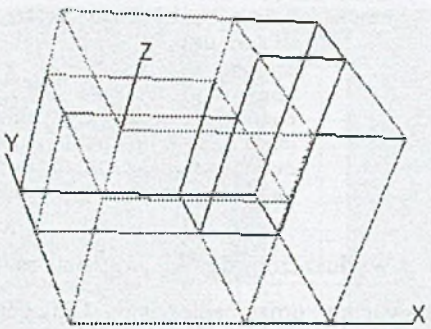
Dane do przykładu 6

Nr prostopadłościanu	1	2	3	4	5
Długości boków	1*2*2	3*2*1	2*2*2	3*1*1	3*1*2

Problem reprezentowany przez przykład 6 może wydawać się jedynie rozszerzeniem problemu z przykładu 2 o trzeci wymiar przestrzeni. W rzeczywistości jest on bardziej skomplikowany, ponieważ dowolna figura może znajdować się w jednej z czterech pozycji przy obrocie wokół jednej osi. Uwzględniając kolejne osie otrzymujemy w rezultacie 64 kombinacje pozycji dla jednej figury. Oczywiście, w przypadku prostopadłościanów niektóre kombinacje pozycji pokrywają się i możemy je zredukować. Rys. 8 przedstawia rozwiązanie przykładu 6 znalezione po $731*10^{-3}$ [s].



Rys.7. Rozwiązanie przykładu 5
Fig.7. Solution for Example 5



Rys.8. Rozwiązanie przykładu 6
Fig.8. Solution for Example 6

Następny przykład wprowadza kolejny stopień trudności w postaci upakowania figur złożonych bez możliwości obrotu.

Przykład 7. Danych jest pięć kątowników przestrzennych o wymiarach podanych w tabeli 7. Należy tak upakować dane figury, aby zajmowały najmniejszą przestrzeń, jednocześnie nie przenikały się.

Tabela 7

Dane do przykładu 7

i	1	2	3	4	5
L	[3,1,1,2,1,3,2,1]	[3,3,2,2,1,3,3,1]	[3,1,3,1,1,3,1,3]	[1,1,2,2,1,2,2,1]	[2,3,1,1,1,2,3,1]

Szczegóły dotyczące sposobu opisu kątowników przestrzennych przez listę długości boków, ideę rozwiązywania tego typu problemów oraz kod źródłowy programu znajdują się w [12]. W przypadku upakowania figur złożonych w przestrzeni bardzo ważnym elementem programu rozwiązującego tego typu problemy jest symetryczność ograniczeń upakowania w poszczególnych płaszczyznach. Nie zachowując tego warunku, otrzymujemy błędne rozwiązanie. Program uwzględniający symetryczność ograniczeń znajduje rozwiązanie po $461*10^{-3}$ [s]. Rozwiązanie to przedstawia rys.9.

Ostatni przykład – najbardziej skomplikowany – przedstawia problem upakowania figur złożonych z możliwością obrotu o kąty będące wielokrotnością kąta 90° we wszystkich trzech osiach.

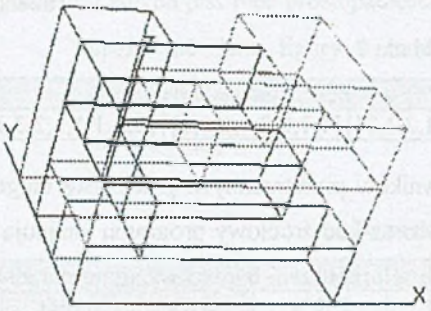
Przykład 8. Danych jest pięć kątowników przestrzennych o wymiarach podanych w tabeli 8. Należy upakować dane figury z możliwością obrotu o wielokrotności kąta 90° we wszystkich trzech osiach, tak aby zajmowały najmniejszą przestrzeń jednocześnie nie przenikając się.

Tabela 8

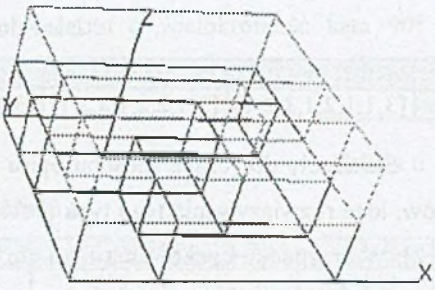
Dane do przykładu 8

i	1	2	3	4	5
L	[2,1,3,1,1,1]	[3,2,3,2,1,1]	[3,1,3,1,1,1]	[2,1,2,1,1,1]	[3,1,2,1,1,1]

Szczegółowy opis listy danych oraz metody rozwiązywania tego typu problemów i pełny kod źródłowy znajdują się w [12]. Porównując rys.9 i rys 10 zauważymy dwa różne rozwiązania pomimo tej samej liczbie kątowników przestrzennych o tych samych wymiarach, z których drugie zajmuje mniejsza przestrzeń upakowania. Jednak rys.9 przedstawia rozwiązanie problemu upakowania kątowników przestrzennych z ustaloną pozycją, co determinuje wymiary w poszczególnych osiach. Rys.10 natomiast przedstawia rozwiązanie problemu otrzymane po 10,035 [s], gdzie kątowniki przestrzenne mogą być obracane przed upakowaniem.



Rys.9. Rozwiązanie przykładu 7
Fig.9. Solution for Example 7



Rys.10. Rozwiązanie przykładu 8
Fig.10. Solution for Example 8

4. Złożoność obliczeniowa problemów upakowania

Prezentowane w poprzednich punktach problemy zaliczamy do zagadnień kombinatorycznych [4, 5, 10]. Jednak wśród zagadnień kombinatorycznych istnieje pewna grupa zagadnień zwanych typu NP (*non-deterministic polynomial*). Zagadnienia typu NP są bardziej „skomplikowane obliczeniowo" od pozostałych zagadnień kombinatorycznych. Problemy upakowania figur na płaszczyźnie i w przestrzeni należą właśnie do zagadnień kombinatorycznych typu NP [8]. Zagadnienia te nie są trudne w opisie, lecz w rozwiązaniu. Rozwiązanie zadań kombinatorycznych w większości przypadków polega na obliczeniu wszystkich możliwych kombinacji, a następnie wybraniu najlepszego rozwiązania spełniającego wszystkie ograniczenia, jednak dla zadań typu NP liczba wszystkich kombinacji może być bardzo duża nawet dla niewielkich wymiarów zadań. Rozwiązanie zadań kombinatorycznych typu NP metodą obliczenia wszystkich możliwych kombinacji i wybrania najlepszej jest - poza najprostszymi przypadkami - nierealne, nawet wykorzystując najszybsze na świecie komputery. Aby zobrazować to obliczmy liczbę możliwych kombinacji dla problemów upakowania figur omawianych w tym artykule:

$$K = N_D^{(N_V * N_F)}$$

gdzie:

K - liczba kombinacji,
N_D - liczba elementów w domenie zmiennych,
N_V - liczba zmiennych opisujących jedną figurę,
N_F - liczba figur.

Tabela 9

Liczby zmiennych opisujące jedną figurę N_V

Figura	2D (na płaszczyźnie)		3D (w przestrzeni)	
	Bez obrotu	z obrotem	bez obrotu	z obrotem
PROSTA	2	4	3	6
ZŁOŻONA	4	8	6	12

W przedstawionych przykładach liczba elementów w domenie zmiennych $N_D = 10$, zaś każdy przykład bazował na pięciu figurach $N_F = 5$. Podstawiając dane do wzoru otrzymujemy wyniki prezentowane w tabeli 10.

Tabela 10

Liczby wszystkich kombinacji K rozwiązań problemów upakowania

Figura	2D (na płaszczyźnie)		3D (w przestrzeni)	
	bez obrotu	z obrotem	bez obrotu	z obrotem
PROSTA	10^{10}	10^{20}	10^{15}	10^{30}
ZŁOŻONA	10^{20}	10^{40}	10^{30}	10^{60}

Porównując liczby wszystkich kombinacji K rozwiązań problemów upakowania figur omawianych w tym artykule z czasami rozwiązań odpowiednich przykładów widzimy, że próba rozwiązywania upakowywania figur prostych i złożonych, na płaszczyźnie i w przestrzeni za pomocą programowania w logice z ograniczeniami (w szczególności za pomocą języka CHIP V.5.2) jest zdumiewająco efektywna.

LITERATURA

1. Aggoun A., Beldiceanu N.: Extending Chip in order to solve complex scheduling and placement problems, Journal Mathematical and Computer Modeling Vol. 17, No. 7, 1993, 57-73.
2. Bryant V.: Aspekty kombinatoryki (Aspects of Combinatorics), WNT, Warszawa 1997.
3. Christofides N., Mingozzi A., Toth P., Sandi C.: Combinatorial Optimization, John Waley & Sons, London 1979.
4. Dincbas M., Simonis H., Van Hentenryck P.: Solving a Cutting-Stock Problem in Constraint Logic Programming, Logic Programming 1, Proc. 5 Inf. Conf. and Symp., MIT Press Cambridge, Massachusetts, London 1998.
5. Elion S., Christofides N.: The loading problem, Management Science Vol. 17, No. 4, 1971, 259-268.
6. Gilmore P.C., Gomory R.E.: The theory and computation of knapsack functions, Operation Research, Vol. 14, 1966, 1045-1079.
7. Jampel M., Freuder E., Maher M.: Over-Constrained Systems , Springer- Verlag Berlin Heidelberg 1996.
8. Lines M.E.: Liczby wokół nas (Think of a Number), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1995.

9. Niederliński A.: Constraint logic programming - from Prolog to Chip, Proceedings of the CPDC'99 Workshop on Constraint Programming for Decision and Control, Gliwice 1999, 27-34.
10. Padberg M.W.: Covering, packing and knapsack problems, Annals of Discrete Mathematics Vol. 4, 1979, 265-287.
11. Szczygieł T.: Solving a two-dimensional angle packing problem in CHIP, Proceedings of the CPDC'99 Workshop on Constraint Programming for Decision and Control, Gliwice 1999, 65-72.
12. Szczygieł T.: Solving a three-dimensional angle packing problem in CHIP, Proceedings of the CPDC'2000 Workshop on Constraint Programming for Decision and Control, Gliwice 2000, 59-66.

Podziękowania

Autor chciałby podziękować Prof. A. Niederlińskiemu za pomoc i wsparcie przy opracowywaniu tego artykułu.

Recenzent: Prof.dr hab.inż. Z.Banaszak

Abstract

The paper is about solving two-dimensional and three-dimensional packing problems for. simple and complex figures. It is well known that figure packing problems belong to NP-hard problems.

The problem discussed has its roots in solving the transportation of high-current enclosed conductors (or high-current enclosed bus bars) in the most economical way. The producer of those bars is interested in packing the largest number of bars to be delivered to the customer in the vehicle space available. The paper aims at presenting CHIP solutions for a number of bar-transportation-related packing problems. For all presented examples the computational complexity has been analyzed.

The paper starts with a discussion and CHIP program for square and rectangle packing problems with no rotation. Next programs solving angle packing problems with rotation are presented. These programs have been generalized for the solution of three-dimensional angle packing problems with rotation.

The effectiveness of the three-dimensional simple figure packing program depends very much upon the symmetry of constraints. The symmetrical constraints are cumulative constraints.

The programs are almost declarative and result in short computation times.