Discrete Mathematics 57 (1985) 105–121
North-Holland

ALL SOLUTIONS OF THE SOMA CUBE PUZZLE

Christoph PETER-ORTH

Scharweg 68, 2300 Kiel 14, Fed. Rep. Germany

Received 10 January 1984
Revised 27 November 1984

Many combinatorial riddles may be translated into integer programming problems. Here the well-known puzzle of the Soma cube is formulated as a set partitioning problem in two different ways. This leads to fast algorithms for the enumeration of all solutions.

1. Introduction

The Soma cube was invented by the Danish writer Piet Hein during a lecture on quantum physics by Werner Heisenberg. It is a three-dimensional puzzle, a nice birthday present and a device for testing ones patience.

Piet Hein established that the seven parts in Fig. 1 which consist of $3+6\times4=27$ elementary (1×1) cubes may be put together into one 3×3 cube. The pieces are exactly the seven different non-convex shapes that can be constructed by joining at most four elementary cubes at their faces: it is clear that the smallest non-convex shape requires three small cubes (number 7 in Fig. 1), and parts 1–6 are obtained by attaching one more elementary cube.

The puzzle is to reassemble the seven pieces into the overall (3×3) cube. It is available commercially under the trade name Soma. Many people have a hard time to complete the task. But once you have managed one solution, you will probably wonder whether there are others. With time you may discover more and more different configurations, but then it becomes rather difficult to remember the previous solutions so that you can be certain that you have found a new configuration. So the question as to the number of essentially different Soma configurations arises naturally.

A first enumeration of all solutions was obtained by hand in 1961 by John Horton Conway and M.J.T. Guy "one wet afternoon" when both mathematicians had no more pressing chores at the University of Cambridge. The total of 240 essentially different configurations was later on also verified by computer programs.

In Conway's opinion "for a puzzle the size of Soma, it's an admission of defeat to use a computer. If you find the right way of organizing the material, it should take less time to do the whole thing by hand than it does to program the

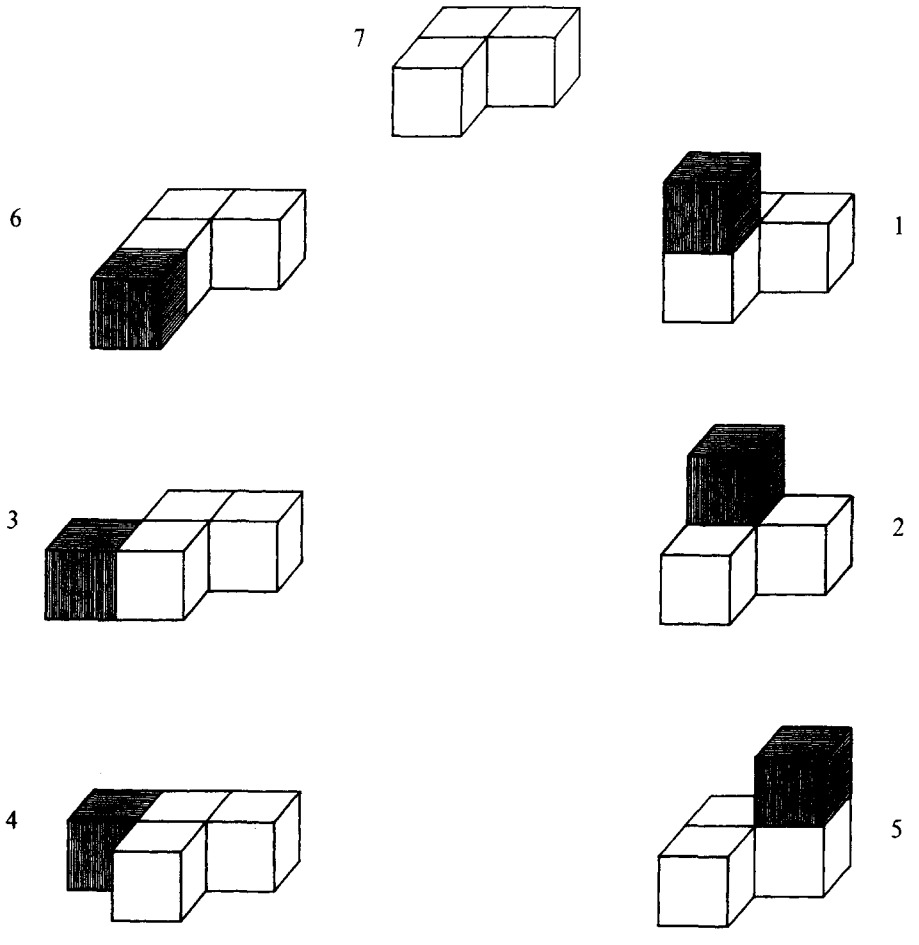

Fig. 1.

machine." Maybe this is valid for people gifted with strong powers of visualization; but others may prefer the transformation to a standard format.

In this article we transform in two different ways. At first we consider the twisted shape 1 as the piece which is to be placed first. With this we need only a few easy geometric preliminaries to arrive at a set partitioning problem whose solutions can be enumerated by a simple program.

For a second approach we start with placing the tripod 2 and the T-shape 4 and put a little more effort into reducing the size of the computer enumeration problem. This also leads to a neat classification of all Soma configurations (11 classes in Table 2) which provides the basis for the complete listing in Table 3.

The classification can be stated well in plain language, but it also leads to an easy program for setting up the matrix on which the final enumeration procedure operates. Finding all (6 to 43) individual Soma configurations within the 11 classes by hand is also quite feasible, but cautious people will prefer the insurance against overlooking any possibilities which a correct computer program provides. A complete listing of the program is given in the Appendix.

2. The first piece as a starting point

The 27 1×1 cubes of the overall 3×3 cube may be divided into four classes which we shall designate by the letters C, F, E, and V:

C. 1 cube in the very center of the 3×3 cube.

F. 6 cubes at the centers of the 6 faces.

E. 12 cubes at the centers of the 12 edges.

V. 8 cubes at the vertices (corners) of the 3×3 cube.

Note that rotations or reflections of the Soma cube do not change the characteristic C, F, E, V of any 1×1 cube.

In Table 1 we list all possible letter combinations for each of the seven shapes. For instance, the first entry FEEV for piece number 1 signifies that it is possible to place the first shape such that it occupies one face center, two edge centers and one vertex (corner) of the 3×3 cube. For a fixed corner one has the first three

Table 1

Piece	Number of possibilities	Characteristics of the pieces
1	96 = 24	FEEV
	24	CFEV
	24	FFEE
	24	CFFE
2	64 = 8	EEEV
	24	FFEV
	24	CFEE
	8	CFFF
3	72 = 48	FEEV
	24	CFFE
4	72 = 24	FEVV
	24	FEEE
	12	CFEE
	12	CFFF
5	96 = 24	FEEV
	24	CFEV
	24	FFEE
	24	CFFE
6	144 = 48	EEVV
	48	FEEV
	24	FFEE
	24	CFFE
7	144 = 24	EEV
	48	FEV
	24	FEE
	12	FFE
	24	CFE
	12	CFF

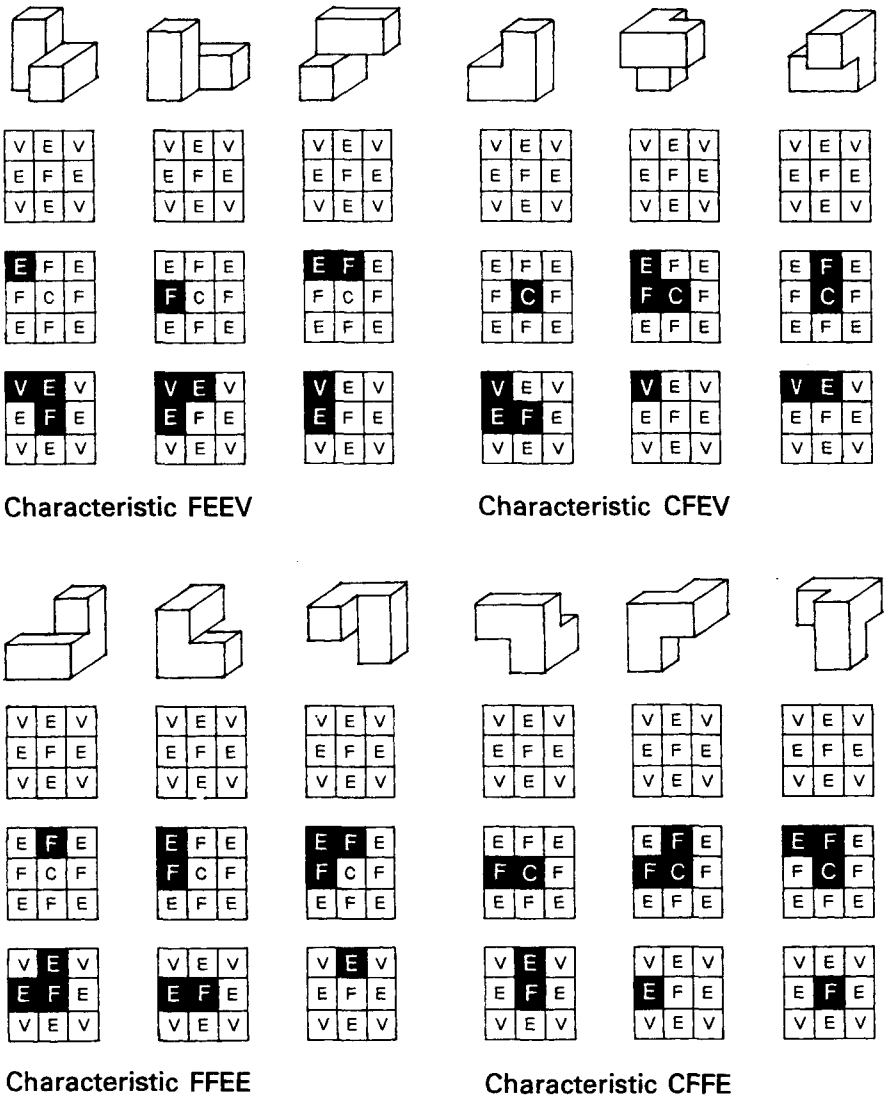

Fig. 2. All possible placements of the first piece within the 2×2 cube at the lower left rear corner of the 3×3 cube.

sketches in Fig. 2, and since there are eight vertices the total number of possibilities FEEV is $3\times8=24$. With patience the reader may verify all other entries in Table 1, but for the following only the first piece has to be examined in detail. This shape (and equally its mirror image number 5) has some easily verified properties:

(i) The first piece is always situated in a uniquely determined 2×2 cube. Obviously, there are 8 possibilities for the enveloping 2×2 cube each containing one of the 8 corners of the 3×3 cube.

(ii) Any given solution of the Soma puzzle may be rotated such that the 2×2 cube around the first shape is turned into the lower left rear corner.

(iii) There are exactly 12 ways to fit the first piece into one 2×2 cube: the three possibilities for each of the characteristics FEEV, CFEV, FFEE, and CFFE in Fig. 2: there the 2×2 cube is placed at the lower left rear vertex of the overall 3×3 cube.

(iv) Positions with different characteristics cannot be equivalent under rotations of the 3×3 cube.

(v) Any placement of the first piece with the same characteristic is equivalent under rotations of the 3×3 cube. To see this rotate the 3×3 cube till the enveloping 2×2 cube of the shape is situated at the lower left rear vertex and define a diagonal from this point to the upper right front corner. Turn it together with the first shape around this diagonal axis by 120 and 240 degrees. This yields a triplet of placements with the same characteristic.

(vi) If the first piece is fixed at one of the four essentially different positions, its unsymmetrical shape prevents the possibility of any rotation of the remaining 23 1×1 cubes.

It follows from the listed properties (i)–(vi) that a complete enumeration of all essentially different solutions may be started by placing the first piece at four positions with different characteristics. Thereafter the other six shapes may be allocated one by one, and no two complete solutions will then be equivalent under the group of the 24 rotations of the entire Soma cube.

3. Formulation as a set partitioning problem

For a complete enumeration we establish a matrix A consisting of zeros and ones. The rows $i = 1, 2, \ldots, 27$ belong to the 27 1×1 cubes. The first 4 columns express the selected essentially different positions of the first piece, and the columns 5 through 596 are assigned to the 592 possible positions of all the other pieces. We add 7 more rows ($i = 28, \ldots, 34$) which mark the shape numbers of the columns. So, if j is a possible position of the piece p occupying the 1×1 cubes s, t, u, and v then $a_{sj} = a_{tj} = a_{uj} = a_{vj} = a_{qj} = 1$ where $q = 27 + p$, and all other a_{ij} are zero.

The matrix A can be generated by a computer, and our task is to determine all feasible solutions of

$$Ax = (1, 1, \ldots, 1)^{\mathrm{T}} \quad (34 \text{ components } 1)$$

$$x_j = 0 \text{ or } 1 \quad \text{for } j = 1, \ldots, 596.$$

This is a 34×596 set partitioning problem without target function. We just have to find all different combinations of columns in the matrix A such that in every row of A a single 1 is selected.

The computer enumeration on a Siemens 7.760 machine (University Kiel) was carried out in a straightforward way. It took 8.4 seconds CPU time to find all 480 solutions which are not equivalent under rotations of the 3×3 Soma cube.

If any solution is reflected in respect to a mirror through the center of the Soma cube, a new valid configuration is obtained which can be rotated into one of the listed solutions. Thus it appears likely that the 480 solutions are in fact 240 pairs (configuration C / mirror image of C), but this is rather hard to check from the

computer listings. In the next section we shall count these pairs directly thereby reducing the overall enumeration effort.

4. Pieces four and two as starting points

The fourth piece has a remarkable property: In any full solution it must be placed in the style FEVV. To see this consider that the shapes 1, 2, 3, 5 and 7 can cover at most one corner of the 3×3 cube whereas the sixth piece may hit two corners. Therefore at least one $(=8-5\times1-2)$ corner has to be reserved for the fourth shape. This excludes the possibilities FEEE, CFEE and CFFF in Table 1 and leaves the two-corner characteristic FEVV as our only choice for the fourth piece.

All other shapes admit more than one of the characteristics in Table 1. However, in any valid Soma configuration the characteristics of all seven pieces must total 1 letter C, 6 letters F, 12 letters E and 8 letters V. With a little bit of patience one can set up the complete list of combinations of characteristics which satisfy this condition. They are the eleven 'branches' in Table 2. It is remarkable that each of the branches admits at least 14 solutions (row a in Table 2). Most people will instinctively place the tripod in one corner, but there are the 74 possibilities of branch one where the tripod (piece 2) covers the center but no vertex of the 3×3 cube.

An enumeration using the eleven branches in Table 2 is faster, it took 0.9 seconds on our Siemens 7.760 computer. For this the fourth piece was fixed inside the bottom layer of the 3×3 cube such that the front edge was covered completely (the fourth 1×1 cube being situated at the center of the bottom layer). With this we could not loose any solution, since the fourth shape must have the characteristic FEVV.

Table 2

	Branch										
Piece	1	2	3	4	5	6	7	8	9	10	11
1	FEEV	FEEV	FEEV	FEEV	CFEV	FEEV	CFEV	FEEV	CFFE	FFEE	CFEV
2	CFEE	EEEV	EEEV	EEEV	EEEV	EEEV	EEEV	EEEV	EEEV	EEEV	EEEV
3	FEEV	CFFE	FEEV	FEEV	FEEV	FEEV	FEEV	FEEV	FEEV	FEEV	FEEV
4	FEVV	FEVV	FEVV	FEVV	FEVV	FEVV	FEVV	FEVV	FEVV	FEVV	FEVV
5	FEEV	FEEV	FEEV	CFEV	FEEV	CFEV	FEEV	CFFE	FEEV	CFEV	FFEE
6	EEVV	EEVV	EEVV	EEVV	EEVV	FEEV	FEEV	EEVV	EEVV	EEVV	EEVV
7	FEV	FEV	CFF	FFE	FFE	FEV	FEV	FEV	FEV	FEV	FEV
a	74	66	38	14	14	21	21	51	51	65	65
l	37	33	19	6	8	9	12	26	25	22	43
r	37	33	19	8	6	12	9	25	26	43	22

Table 3

	611	666	666	337	666	666	666	655	655	115	113	511	155	771	773
top	615	511	551	157	156	615	551	677	633	155	175	517	115	571	511
	665	551	511	155	155	115	511	337	337	773	775	337	773	553	551
	221	226	226	233	233	233	276	615	615	615	613	551	156	611	673
center	327	327	723	227	227	227	223	225	225	223	223	227	223	223	223
	355	351	513	115	115	155	513	233	277	273	255	233	273	253	251
	327	327	723	666	337	337	773	611	611	666	666	666	666	666	666
bottom	347	347	743	246	247	247	243	241	241	243	243	246	243	243	243
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
	663	663	666	625	723	773	336	655	655	666	666	776	551	116	116
top	511	511	511	655	715	215	233	277	233	277	233	276	251	776	776
	551	551	551	677	115	115	711	337	337	337	337	336	773	723	726
	633	673	633	225	223	273	256	615	615	116	116	116	511	156	156
center	227	223	227	123	723	223	221	225	225	225	225	225	223	123	123
	251	251	251	673	155	155	771	233	277	233	277	233	273	223	223
	637	677	337	113	666	666	556	611	611	155	155	155	666	556	553
bottom	247	243	247	143	643	643	546	641	641	145	145	145	643	543	543
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
	551	666	667	116	116	661	611	115	116	611	611	711	666	776	666
top	751	311	225	226	226	221	221	155	136	615	671	771	156	176	617
	723	321	325	723	726	327	327	666	557	665	677	666	155	116	117
	511	356	617	156	156	611	655	215	216	231	255	255	233	255	255
center	723	327	327	123	123	325	321	337	533	337	331	331	337	133	337
	223	221	355	773	773	377	377	633	577	355	633	633	115	336	133
	666	557	611	556	553	655	665	227	226	227	225	225	227	225	225
bottom	643	547	341	543	543	345	345	247	243	247	245	245	247	245	245
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
	666	666	666	225	226	227	221	226	226	226	655	655	556	667	666
top	615	511	633	255	276	277	231	217	216	233	277	233	256	235	277
	115	551	711	666	776	666	557	117	116	711	227	227	226	225	227
	233	236	255	215	255	255	211	256	256	256	615	615	513	617	116
center	337	337	331	733	133	133	533	337	337	331	335	335	733	337	335
	155	351	771	336	336	336	577	133	133	771	233	277	236	255	233
	227	227	225	711	115	115	666	556	557	556	611	611	711	611	155
bottom	247	247	245	741	145	145	643	546	547	546	641	641	741	341	145
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
	666	776	776	766	666	116	117	661	611	666	115	116	116	551	311
top	233	276	275	235	235	236	235	231	231	211	155	176	176	571	371
	227	226	225	225	225	227	225	227	227	221	666	553	556	666	666
	116	116	336	776	677	156	177	611	655	356	215	216	216	211	255
center	335	335	133	133	133	133	133	335	331	337	773	573	573	573	371
	277	233	255	255	255	277	255	277	277	231	673	573	573	673	376
	155	155	116	116	117	556	666	655	665	557	223	226	223	223	225
bottom	145	145	146	143	143	543	643	345	345	547	243	243	243	243	245
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444

Table 3 *cont.*

	666	225	223	221	221	221	226	226	223	116	116	113	311	551	116
top	551	255	273	271	271	251	216	216	215	276	276	275	215	251	156
	511	666	666	666	553	355	113	116	115	223	226	225	225	223	553
	276	215	255	211	211	211	256	256	273	156	156	173	371	511	216
center	773	377	173	375	573	377	773	773	773	173	173	173	377	773	753
	513	376	673	376	573	375	173	173	155	273	273	255	255	273	773
	223	311	115	355	666	666	556	553	666	556	553	666	666	666	226
bottom	243	341	145	345	643	346	543	543	643	543	543	643	346	643	243
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	, 444
	116	116	226	221	221	113	666	366	663	226	226	371	113	761	335
top	156	133	256	251	271	613	336	336	613	276	276	275	215	761	133
	556	335	556	553	375	666	133	136	611	173	176	225	225	566	116
	216	216	216	211	211	255	255	255	255	256	256	371	773	211	255
center	753	755	357	753	355	713	117	117	713	113	113	311	713	753	157
	773	775	377	773	375	773	177	177	773	173	173	255	255	553	666
	223	226	311	666	666	225	225	225	225	556	553	666	666	223	227
bottom	243	246	341	643	346	245	245	245	245	543	543	346	643	243	247
	444	444	444	444	444	, 444	444	444	444	444	444	444	444	, 444	444
	223	227	223	226	225	223	311	557	336	367	163	136	367	736	336
top	233	233	233	255	233	216	216	533	666	365	163	156	367	776	666
	736	336	536	775	336	117	227	336	557	665	766	755	661	551	551
	216	217	216	216	255	233	331	277	233	277	255	233	255	233	233
center	556	557	756	356	157	556	556	511	511	311	113	116	317	516	517
	776	666	556	376	666	177	277	666	577	355	773	775	311	511	511
	511	511	711	311	117	536	536	221	221	221	225	223	225	223	227
bottom	541	541	741	341	147	546	546	241	241	241	245	246	245	246	247
	444	444	444	444	444	444	444	, 444	444	444	444	444	444	444	444
	365	336	553	365	116	116	116	116	661	651	366	356	336	667	657
top	255	233	256	255	157	156	156	156	651	651	311	311	533	651	651
	227	221	221	221	337	773	776	336	677	677	771	771	711	611	611
	365	666	533	365	216	216	216	216	211	211	256	256	256	277	277
center	311	715	716	317	557	553	553	557	553	553	355	355	551	553	553
	277	211	211	211	533	573	573	533	573	673	371	371	771	513	613
	661	755	736	667	226	226	223	227	223	223	226	226	226	223	223
bottom	341	745	746	347	246	243	243	247	243	243	245	246	246	243	243
	444	444	444	444	, 444	444	444	444	444	444	444	444	444	444	444
	226	227	221	221	226	226	226	226	666	113	771	773	663	663	666
top	256	256	251	251	216	216	216	211	253	255	271	211	211	211	211
	776	666	773	357	113	116	116	771	223	227	223	221	221	221	221
	215	217	211	211	276	276	233	256	613	153	611	673	633	673	633
center	355	357	553	355	553	553	557	355	753	153	553	553	557	553	557
	376	355	573	377	153	153	156	371	255	277	253	251	251	251	251
	311	311	666	666	776	773	337	366	711	666	666	666	637	677	337
bottom	341	341	643	346	543	543	547	345	741	643	543	543	547	543	547
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444 ,

Table 3 *cont.*

	736	615	773	775	556	557	666	366	316	557	666	226	225	226	226
top	776	655	173	155	516	516	716	316	316	511	611	275	255	256	216
	556	677	666	666	766	666	733	776	776	666	553	775	666	355	776
	233	215	255	275	211	277	255	255	255	277	277	216	275	276	211
center	511	113	113	113	513	513	711	311	311	513	513	311	311	311	315
	516	673	613	613	773	113	331	371	376	613	513	355	316	315	376
	223	223	225	223	223	223	225	225	225	223	227	316	377	776	355
bottom	241	243	245	243	243	243	245	245	245	243	243	346	341	341	345
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
	227	227	226	226	773	773	317	316	316	766	556	556	556	556	556
top	211	211	216	216	275	215	215	215	215	211	117	116	116	116	331
	666	553	713	716	225	225	225	225	225	223	337	773	776	336	733
	277	277	256	256	613	173	377	336	376	756	256	256	256	256	256
center	315	513	113	113	113	113	311	711	311	713	157	153	153	157	751
	316	513	773	773	255	255	255	255	255	213	133	173	173	133	711
	355	666	556	553	666	666	666	736	776	556	226	226	223	227	226
bottom	345	643	543	543	143	643	346	746	346	543	246	243	243	247	246
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
	316	556	556	665	667	556	556	225	226	226	226	227	226	225	225
top	311	776	776	677	655	116	116	233	256	256	256	233	233	271	217
	775	713	716	617	615	713	716	337	553	556	556	335	335	377	317
	216	256	256	255	277	256	256	255	276	276	233	277	276	255	255
center	355	153	153	153	153	753	753	157	153	153	157	155	155	351	357
	375	113	113	113	113	713	713	117	113	113	116	115	115	311	311
	226	226	223	223	223	226	223	666	776	773	337	666	776	666	666
bottom	246	243	243	243	243	243	243	146	143	143	147	146	146	346	346
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
	225	513	666	366	366	377	666	666	666	666	336	337	337	356	336
top	211	211	336	376	336	376	576	756	556	655	576	577	533	356	756
	317	227	733	776	736	666	773	753	753	753	776	666	666	776	756
	255	513	211	211	211	211	211	211	211	211	233	233	277	255	233
center	351	553	715	315	715	315	513	713	513	713	511	511	511	311	711
	377	277	755	355	755	355	553	553	773	753	556	556	556	376	556
	666	666	221	221	221	221	221	221	221	221	221	221	221	221	221
bottom	346	643	245	245	245	245	243	243	243	243	241	241	241	241	241
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
	336	137	156	117	731	331	776	666	366	736	666	356	226	226	226
top	556	177	156	516	711	511	336	336	336	776	655	356	276	255	277
	756	666	766	666	666	666	133	133	136	166	153	116	377	377	357
	233	233	255	277	233	233	276	277	277	233	277	255	216	216	216
center	511	115	113	513	715	517	115	115	115	115	113	317	315	315	315
	776	655	773	553	655	556	155	155	155	155	153	316	355	375	355
	221	223	223	223	223	227	226	227	227	223	227	227	116	116	116
bottom	241	245	243	243	245	247	245	245	245	245	243	247	345	346	346
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444

Table 3 *cont.*

	226	227	226	366	356	136	331	311	116	337	357	377	357	356	356
top	255	255	216	277	255	233	233	277	216	233	257	275	255	255	255
	357	153	311	227	227	223	227	227	226	221	221	221	221	221	221
	216	277	276	316	316	116	551	316	336	557	355	355	377	336	376
center	317	113	315	315	315	715	511	315	715	517	317	315	315	715	315
	357	153	355	255	277	255	277	255	255	211	211	211	211	211	211
	116	666	776	116	116	766	666	666	733	666	666	666	666	736	776
bottom	346	643	345	345	346	745	546	345	745	546	346	346	346	746	346
	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444

Having fixed the fourth piece we examined all possibilities of placing the second shape (tripod) with the characteristics EEEV and CFEE. These can be divided into two classes in respect to the vertical reflecting plane through the center. Reflection by this mirror leaves our fixed shape 4 invariant, but any placement of the tripod has a different mirror image: the tripod must touch either the left face (l) or the right face (r) of the overall cube. There are three positions each with EEEV-l and EEEV-r and nine possible placements each with CFEE-l and CFEE-r. These were found by trial and error and confirmed by a separate computer enumeration.

With the fourth piece fixed as described and with the second piece placed in l-positions only, we obtained eleven matrices one for each of the possible combinations of characteristics in Table 2. This led to the fast enumeration of the 240 solutions in the final Table 3 and proved that the 480 solutions are in fact evenly divided into 240 l configurations and their 240 mirror images (r). Each r solution is obtained from a listed l configuration by simply exchanging the left and right columns.

Acknowledgment

The author would like to thank a referee for his useful comments and also Dr. J.H. Ahrens whose assistance improved the clarity of the formulations.

Appendix

```
****  SOURCE  LISTING  ****      SIEMENS  FORTRAN  COMPILER   F O R  1     V1.50          PAGE

                                                PROGRAM UNIT: MAIN

DO/IF SEG    STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------------------
      1/1       1          1 |      PROGRAM MAIN
                           2 |C
                           3 |C   The subroutine      The subroutine PUZZLE operates on special
                           4 |C   SOMA forms the      parts of this structure
                           5 |C   data structure
                           6 |C                    matrices:
                           7 |C   shape              1 - 11        1 - 7 (2)     1        1 - 7       1,2
                           8 |C
                           9 |C   4 FEVV   1      XXXXXXXXXXX       XXXXXXX       X       XXXXXXX       XX
                          10 |C
                          11 |C   2 CFEEl  9      X::::::::::       X::::::       X       X::::::       X:
                          12 |C     EEEVl  3      :XXXXXXXXXX       :XXXXXX       X       :XXXXXX       XX
                          13 |C     EEEVr  3      ::::::::::        :::::::       :       :::XXXX       :X
                          14 |C
                          15 |C   1 FFEE   15      :::::::::X:       ::::::X       X       ::::::X       ::
                          16 |C     CFEV   16      ::::X:X:::X       :::XX:X       X       :::XX::       ::
                          17 |C     FEEV   16      XXXX:X:X:::       XXXXXX:       X       XXX::X:       X:
                          18 |C     CFFE   15      :::::::::X::      :::::X:       X       :::::::       ::
                          19 |C
                          20 |C   5 FFEE   15      ::::::::::X       ::::::X       X       :::::::       ::
                          21 |C     CFEV   16      :::X:X:::X:       :::XX:X       X       ::::::X       ::
                          22 |C     FEEV   16      XXX:X:X:X::       XXXXXX:       X       XXXXX::       X:
                          23 |C     CFFE   15      :::::::X:::       :::::X:       X       :::::X:       ::
                          24 |C
                          25 |C   6 EEVV   28      XXXXX::XXXX       XXXX:XX       X       XXXX:XX       XX
                          26 |C     FEEV   30      :::::XX::::       ::::X::       X       ::::X::       :X
                          27 |C
                          28 |C   3 FEEV   30      X:XXXXXXXXX       X:XXXXX       X       X:XXXXX       XX
                          29 |C     CFFE   15      :X:::::::::       :X:::::       X       :X:::::       X:
                          30 |C
                          31 |C   7 FFE     8      :::XX::::::       :::X:::       X       :::X:::       :X
                          32 |C     FEV    32      XX:::XXXXXX       XX::XXX       X       XX::XXX       XX
                          33 |C     CFF     8      ::X:::::::        ::X::::       X       ::X::::       X:
                          34 |C
                          35 |C     sum: 291                                  1. shape: (FFEE u FEEV)   :X
                          36 |C   (sum of all columns)                       5. shape: (CFEV u CFFE)   :X
                          37 |C
        1        2        38 |       REAL TIME,ZEIT,ARRAY T(11)
        1        3        39 |       INTEGER BITI,BIT(28),H(21),T(21),ONE,STEER(28),BLOCK(2:7),MAX
        1        4        40 |       INTEGER PIECE(960),HEAD(23),TAIL(22),NR(2:7),RESULT(240,7),COUNT
        1        5        41 |       INTEGER ROW,COLUMN,SUM,PROD,SHAPE,MARK,FIELD(27,12),KT,LT
        1        6        42 |       INTEGER ORDER(2:7,28),NO(2:7),HH(2:7,28),TT(2:7,28),HT,CHOOSE
        1        7        43 |       INTEGER INDEX,I,J,K,L,NUMBER,MATRIX,ARRAY M(11),ARRAY C(11)
        1        8        44 |       LOGICAL EMPTY
        1        9        45 |       CHARACTER C
        1                 46 |
                          47 |C      Initiate all possibilities for testing different matrices
        1                 48 |
        1       10        49 |       DATA ONE,NO,(RESULT(I,4),I=1,240) /1,2,1,5,6,3,7,240*464/
        1       11        50 |       DATA BLOCK /7,262264,526208,6144,24576,229376/
        1       12        51 |       DATA STEER /195787,223779,899078,76321,84514,141858,43826,78642,
        1                 52 |      ,77410,76186,76321,84514,141858,43298,43538,78114,78354,76834,
        1                 53 |      ,76354,76042,75922,76321,84514,141858,43542,78358,76838,76046/
        1                 54 |
        1       13        55 |       CALL SOMA(PIECE,H,T,BIT)
        1       14        56 |       INDEX=H(20)
        2       15        57 |       DO 1 I=H(5),T(5)
  1     3       16        58 |          PIECE(INDEX)=PIECE(I)
  1     3       17        59 |          INDEX=INDEX+1
  1     4       18        60 |     1 CONTINUE
        5       19        61 |       DO 2 I=H(7),T(7)
  1     6       20        62 |          PIECE(INDEX)=PIECE(I)
  1     6       21        63 |          INDEX=INDEX+1
  1     7       22        64 |     2 CONTINUE
        7       23        65 |       T(20)=INDEX-1
        7       24        66 |       H(21)=INDEX
        8       25        67 |       DO 3 I=H(10),T(10)
  1     9       26        68 |          PIECE(INDEX)=PIECE(I)
  1     9       27        69 |          INDEX=INDEX+1
  1    10       28        70 |     3 CONTINUE
       11       29        71 |       DO 4 I=H(12),T(12)
  1    12       30        72 |          PIECE(INDEX)=PIECE(I)
  1    12       31        73 |          INDEX=INDEX+1
  1    13       32        74 |     4 CONTINUE
       13       33        75 |       T(21)=INDEX-1
       13       34        76 |       HEAD(8)=INDEX
       14       35        77 |       DO 7 HT=1,28
  1    15       36        78 |          INDEX=STEER(HT)
  1    16       37        79 |          DO 6 K=2,7
  2    17       38        80 |             ORDER(K,HT)=K
```

```
DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-----------------------------------------------------------------------------------------
  2    17     39        81 |            SHAPE=BLOCK(K)
  2    17     40        82 |            I=IAND(SHAPE,INDEX)
  2    17     41        83 |            EMPTY=.TRUE.
  2    18     42        84 |            DO 5 J=2,21
  3    19     43        85 |               IF (IAND(I,ONE) .EQ. 1) THEN
  4    20     44        86 |                  IF (EMPTY) THEN
  5    21     45        87 |                     HH(K,HT)=H(J)
  5    21     46        88 |                     EMPTY=.FALSE.
  5    21     47        89 |                  END IF
  4    22     48        90 |                  IF (I .EQ. 1) THEN
  5    23     49        91 |                     TT(K,HT)=T(J)
  5    23     50        92 |                     GO TO 6
  5    23     51        93 |                  END IF
  4    23     52        94 |               END IF
  3    24     53        95 |               I=ISRL(I,ONE)
  3    24     54        96 |     5       CONTINUE
  2    25     55        97 |     6    CONTINUE
  1    26     56        98 |     7 CONTINUE
       27     57        99 |       OPEN (9,FILE='#SOMA-CUBE.WITHOUT.REFLECTIONS')
       27               100 |
                        101 |C      Main part
       27               102 |
       28     58       103 |     8 WRITE (2,*) 'How many matrices ? Choose  1 , 2 , 7 or 11 !'
       28     59       104 |       READ (1,*) CHOOSE
       28     60       105 |       HT=CHOOSE
       28     61       106 |       IF (HT.NE. 1 .AND. HT.NE. 2 .AND. HT.NE. 7 .AND. HT.NE. 11) STOP
       30     63       107 |       IF (CHOOSE .EQ. 7) THEN
  1    31     64       108 |          WRITE (2,*) 'Selection only with the 2. piece ?  Type:  2 !'
  1    31     65       109 |          READ (1,'(A)') C
  1    31     66       110 |          IF (C .EQ. '2') THEN
  2    32     67       111 |             HT=4
  2    32     68       112 |          ELSE
  2    33     69       113 |             HT=22
  2    33     70       114 |          END IF
  1    33     71       115 |       END IF
       34     72       116 |       WRITE (2,*)'Do you want to change the order of the blocks ? Y=yes'
       34     73       117 |       READ (1,'(A)') C
       34     74       118 |       IF (C .EQ. 'y'  .OR.  C .EQ. 'Y') THEN
  1    35     75       119 |          K=1
  1    36     76       120 |          DO 11 J=HT,HT+CHOOSE-1
  2    37     77       121 |             WRITE (2,'(30X,I2,A)') K,'. matrix:'
  2    40     78       122 |             WRITE (2,'(A,6I3)') ' Length of the blocks 2 - 7 :',
  2    40              123 |                (TT(ORDER(I,J),J)-HH(ORDER(I,J),J)+1,I=2,7)
  2    43     79       124 |             WRITE (2,'(A,6I3,A)') '  Order of the blocks 2 - 7 :',
  2    43              125 |                (ORDER(I,J),I=2,7),'  and the new order shall be:'
  2    47     80       126 |             READ (1,*) (ORDER(I,J),I=2,7)
  2    47     81       127 |             SUM=0
  2    47     82       128 |             PROD=1
  2    48     83       129 |             DO 10 I=2,7
  3    49     84       130 |                SUM=SUM+ORDER(I,J)
  3    49     85       131 |                PROD=PROD*ORDER(I,J)
  3    50     86       132 |    10       CONTINUE
  2    51     87       133 |             IF (SUM .NE. 27  .OR.  PROD .NE. 5040) GO TO 9
  2    52     89       134 |             K=K+1
  2    52     90       135 |    11    CONTINUE
  1    52     91       136 |       END IF
       53     92       137 |       ZEIT=0
       53     93       138 |       SUM=0
       53     94       139 |       COUNT=0
       54     95       140 |       DO 13 MATRIX=1,CHOOSE
  1    56     96       141 |          DO 12 I=2,7
  2    57     97       142 |             HEAD(I)=HH(ORDER(I,HT),HT)
  2    57     98       143 |             TAIL(I)=TT(ORDER(I,HT),HT)
  2    57     99       144 |             NR(I)=NO(ORDER(I,HT))
  2    57    100       145 |    12    CONTINUE
  1    58    101       146 |          HT=HT+1
  1    58    102       147 |          MAX=0
  1    58    103       148 |          TIME=0
  1    58    104       149 |          KT=KTIME(KT)
  1    58    105       150 |          CALL PUZZLE(PIECE,HEAD,TAIL,NR,RESULT,COUNT,MAX)
  1    58    106       151 |          TIME=(KTIME(LT)-KT)/10000.0
  1    58    107       152 |          ZEIT=ZEIT+TIME
  1    58    108       153 |          NUMBER=COUNT-SUM
  1    58    109       154 |          SUM=COUNT
  1    58    110       155 |          ARRAY T(MATRIX)=TIME
  1    58    111       156 |          ARRAY C(MATRIX)=NUMBER
  1    58    112       157 |          ARRAY M(MATRIX)=MAX
  1    58    113       158 |          WRITE (2,'(I5,F12.4,2I5)') MATRIX,TIME,NUMBER,MAX
  1    58    114       159 |    13 CONTINUE
       59    115       160 |       WRITE (2,'(A,F8.4,A,I9,A)') ' Time:',ZEIT,' s.',SUM,' solutions'
       59    116       161 |       WRITE (2,*) 'Do you wish an output of all solutions ? Y=yes'
       59    117       162 |       READ (1,'(A)') C
       59    118       163 |       IF (C .EQ. 'Y'  .OR.  C .EQ. 'y') THEN
       59              164 |
                        165 |C      Evaluation of all solutions
       59               166 |
  1    60    119       167 |          ROW=6
  1    61    120       168 |          DO 19 COUNT=1,229,12
  2    62    121       169 |             COLUMN=0
  2    63    122       170 |             DO 16 MARK=COUNT,COUNT+11
```

```
 DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------
   3    64    123       171 |               COLUMN=COLUMN+1
   3    65    124       172 |               DO 15 J=1,7
   4    66    125       173 |                  SHAPE=RESULT(MARK,J)
   4    66              174 |
                        175 |C     Transformation of the binary data into the numbered pieces
   4    66              176 |
   4    67    126       177 |                  DO 14 I=1,27
   5    68    127       178 |                     BITI=BIT(I)
   5    68    128       179 |                     IF (IAND(SHAPE,BITI) .EQ. BITI) FIELD(I,COLUMN)=J
   5    71    130       180 |   14              CONTINUE
   4    71    131       181 |   15           CONTINUE
   3    72    132       182 |   16        CONTINUE
   3    72              183 |
                        184 |C     Write the solutions on file #SOMA-CUBE.WITHOUT.REFLECTIONS
   3    72              185 |
   2    73    133       186 |            IF (ROW .EQ. 6) THEN
   3    74    134       187 |               WRITE (9,'(''A '')')
   3    74    135       188 |               ROW=1
   3    74    136       189 |            END IF
   2    76    137       190 |            DO 18 L=19,1,-9
   3    78    138       191 |               DO 17 K=L,L+6,3
   4    85    139       192 |                  WRITE (9,'(27X,12(3X,3I1))')
   4    85              193 |     :            ((FIELD(I,J),I=K,K+2),J=1,12)
   4    85    140       194 |   17             CONTINUE
   3    86    141       195 |               WRITE (9,'('' '')')
   3    86    142       196 |   18        CONTINUE
   2    87    143       197 |            WRITE (9,'('' '')')
   2    87    144       198 |            WRITE (9,'('' '')')
   2    87    145       199 |            ROW=ROW+1
   2    87    146       200 |   19     CONTINUE
   1    88    147       201 |         WRITE (9,'(''A '')')
   1    88    148       202 |      END IF
        89    149       203 |      WRITE (9,'('' '')')
        89    150       204 |      WRITE (9,'('' '')')
        89    151       205 |      WRITE (9,'(30X,A,13X,A)') 'matrix  number  time',
        89              206 |     ,'order and length (blocks 2 - 7)  max'
        89    152       207 |      WRITE (9,'('' '')')
        89    153       208 |      K=1
        90    154       209 |      DO 20 J=HT-CHOOSE,HT-1
   1    97    155       210 |         WRITE (9,'(27X,2I8,F9.4,A,6I2,3X,6I3,I7)') K,ARRAY C(K),
   1    97              211 |     ,      ARRAY T(K),' sec.  ',(ORDER(I,J),I=2,7),
   1    97              212 |     ,      (TT(ORDER(I,J),J)-HH(ORDER(I,J),J)+1,I=2,7),ARRAY M(K)
   1    97    156       213 |         K=K+1
   1    97    157       214 |   20 CONTINUE
        98    158       215 |      WRITE (9,'('' '')')
        98    159       216 |      WRITE (9,'(33X,A,I6,F9.4,A)') 'sum:',SUM,ZEIT,' sec.'
        98    160       217 |      GO TO 8
        98    161       218 |      END

****  SOURCE  LISTING  ****      SIEMENS  FORTRAN  COMPILER   F C R  1     V1.50          PAGE     1

                                                PROGRAM UNIT: SOMA

 DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------

    1/1     1         1 |      SUBROUTINE SOMA(PIECE,H,T,BIT)
    1/1               2 |
      1     2         3 |      INTEGER BITI,BIT(28),CENTER,FACE LR,FACE TB,FACE FR,EDGE,VERTEX
      1     3         4 |      INTEGER ROTATE(0:1,27),BOTTOM(2),DOWN,BACK(2),AWAY,SIDE(2),LEFT
      1     4         5 |      INTEGER DEF(7),ALLPOS(24),ESSENZ(24),A(300),PIECE(960),NEW,CUBE
      1     5         6 |      INTEGER SHAPE,SHAPE4,CREATE,STEER,ONE,SHIFT,ZERONE,KIND,TYPE(2:19)
      1     6         7 |      INTEGER HEAD,H(21),TAIL,T(21),LIST,NUMBER,INDEX,I,J,K,JJ(2:7)
      1               9 |
                       10 |C     Initialisierung
      1              11 |
      1     8        12 |      DATA CREATE,ONE,JJ /612501,1,3,4,4,2,2,3/
      1     9        13 |      DATA TYPE /1,2,2,7,2,5,4,7,2,5,4,2,5,5,4,7,5,3/
      1    10        14 |      DATA DEF /464,523,531,537,15,30,11/
                       15 |C        Shape:  4   2   1   5   6  3  7
      1              16 |
                       17 |C     !! Dieses Unterprogramm benutzt die Funktionen:
                       18 |C     !! IAND (=logisches UND)
                       19 |C     !! IOR  (=logisches ODER)
                       20 |C     !! IXOR (=logisches EXCLUSIV ODER)
                       21 |C     !! ISLL (=logisches Bitverschieben nach links)
                       22 |C     !! ISRL (=logisches Bitverschieben nach rechts)
      1              23 |
      1    11        24 |      BIT(1)=ONE
      1    12        25 |      BITI=BIT(1)
      2    13        26 |      DO 11 I=2,28
 1    3    14        27 |         BIT(I)=ISLL(BITI,ONE)
 1    3    15        28 |         BITI=BIT(I)
 1    4    16        29 |   11 CONTINUE
 1    4              30 |
                       31 |C     Steuerparameter: plan drehen (um die senkrechte Achse nach rechts)
 1    4              32 |
      4    17        33 |      INDEX=1
      5    18        34 |      DO 44 K=3,21,9
 1    7    19        35 |         DO 33 J=K,K-2,-1
 2    9    20        36 |            DO 22 I=J,J+6,3
```

```
DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------

 3    10     21        37 |              ROTATE(0,I)=BIT(INDEX)
 3    10     22        38 |              INDEX=INDEX+1
 3    11     23        39 |   22       CONTINUE
 2    11     24        40 |   33    CONTINUE
 1    12     25        41 |   44 CONTINUE
 1    12               42 |
                       43 |C     Steuerparameter: diagonal drehen (unten, hinten, links - Mitte)
 1    12               44 |
      13     26        45 |      INDEX=1
      14     27        46 |      DO 66 K=1,9
 1    16     28        47 |         DO 55 I=K,K+18,9
 2    17     29        48 |            ROTATE(1,I)=BIT(INDEX)
 2    17     30        49 |            INDEX=INDEX+1
 2    18     31        50 |   55    CONTINUE
 1    18     32        51 |   66 CONTINUE
 1    18               52 |
                       53 |C     Steuerparameter: schieben
 1    18               54 |
      19     33        55 |      BITI=BIT(28)-1
      19               56 |
                       57 |C     nach oben/unten
      19               58 |
      19     34        59 |      BOTTOM(1)=7*73
      19     35        60 |      BOTTOM(2)=BOTTOM(1)*513
      19     36        61 |      DOWN=BITI-BOTTOM(1)
      19               62 |
                       63 |C     nach vorne/hinten
      19               64 |
      19     37        65 |      BACK(1)=7*262657
      19     38        66 |      BACK(2)=BACK(1)*9
      19     39        67 |      AWAY=BITI-BACK(1)
      19               68 |
                       69 |C     nach rechts/links
      19               70 |
      19     40        71 |      SIDE(1)=73*262657
      19     41        72 |      SIDE(2)=SIDE(1)*3
      19     42        73 |      LEFT=BITI-SIDE(1)
      19               74 |
                       75 |C     Steuerparameter: sortieren
      19               76 |
      19     43        77 |      CENTER=BIT(14)
      19     44        78 |      FACE LR=BIT(13)+BIT(15)
      19     45        79 |      FACE TB=BIT(23)+BIT(5)
      19     46        80 |      FACE FR=BIT(17)+BIT(11)
      19     47        81 |      VERTEX=BIT(1)+BIT(3)+BIT(7)+BIT(9)+BIT(19)+BIT(21)+BIT(25)+BIT(27)
      19     48        82 |      EDGE=BITI - CENTER - VERTEX - FACE LR - FACE TB - FACE FR
      19               83 |
      19     49        84 |      H(1)=1
      19     50        85 |      SHAPE4=DEF(1)
      19     51        86 |      PIECE(1)=SHAPE4
      19     52        87 |      T(1)=1
      19     53        88 |      H(2)=2
      19     54        89 |      LIST=2
      19     55        90 |      HEAD=2
      19     56        91 |      INDEX=2
      19               92 |
      19               93 |
      20     57        94 |      DO 9 FIGURE=2,7
      20               95 |
                       96 |C     Das definierte Teil wird im Raum gedreht ...
      20               97 |
 1    21     58        98 |         ALLPOS(1)=DEF(FIGURE)
 1    21     59        99 |         STEER=CREATE
 1    22     60       100 |         DO 3 K=1,23
 2    22     61       101 |            J=K+1
 2    23     62       102 |            SHAPE=ALLPOS(K)
 2    23     63       103 |            NEW=0
 2    23     64       104 |            ZERONE=IAND(STEER,ONE)
 2    24     65       105 |            DO 1 I=1,27
 3    25     66       106 |               BITI=BIT(I)
 3    25     67       107 |               IF (IAND(SHAPE,BITI) .EQ. BITI) THEN
 4    26     68       108 |                  CUBE=ROTATE(ZERONE,I)
 4    26     69       109 |                  NEW=IOR(NEW,CUBE)
 4    26     70       110 |               END IF
 3    28     71       111 |    1       CONTINUE
 3    28               112 |
                       113 |C     ... und in die unterste Ecke (unten, hinten, links) geschoben
 3    28               114 |
 2    28     72       115 |            DO 2 I=1,2
 3    29     73       116 |               SHIFT=0
 3    29     74       117 |               IF (IAND(NEW,DOWN) .EQ. NEW) SHIFT=SHIFT+9
 3    31     76       118 |               IF (IAND(NEW,AWAY) .EQ. NEW) SHIFT=SHIFT+3
 3    33     78       119 |               IF (IAND(NEW,LEFT) .EQ. NEW) SHIFT=SHIFT+1
 3    35     80       120 |               IF (SHIFT .GT. 0) NEW=ISRL(NEW,SHIFT)
 3    38     82       121 |    2       CONTINUE
 2    38     83       122 |            ALLPOS(J)=NEW
 2    38     84       123 |            STEER=ISRL(STEER,ONE)
 2    38     85       124 |    3    CONTINUE
 2    38               125 |
                       126 |C     Bestimmung jeder denkbaren Lage des Teils im Soma-Cube
```

```
 DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------

  2    38             127 |
  1    39     86      128 |          NUMBER=24
  1    39     87      129 |          CALL ORDNE(ALLFOS,ESSENZ,ONE,NUMBER)
  1    39     88      130 |          DO 8 J=1,JJ(FIGURE)
  2    41     89      131 |             DO 7 K=1,NUMBER
  3    42     90      132 |                CUBE=ESSENZ(K)
  3    42     91      133 |                TOP=MOTION(CUBE,BOTTOM)*9
  3    42     92      134 |                FRONT=MOTION(CUBE,BACK)*3
  3    42     93      135 |                R SIDE=MOTION(CUBE,SIDE)
  3    42     94      136 |                L SIDE=0
  3    42     95      137 |                IF (FIGURE .EQ. 2) THEN
  4    43     96      138 |                   IF (J .LT. 3) R SIDE=0
  4    45     98      139 |                   IF (J .EQ. 3) L SIDE=1
  4    46    100      140 |                END IF
  3    47    101      141 |                DO 6 UF=C,TOP,9
  4    48    102      142 |                   DO 5 FORW=0,FRONT,3
  5    49    103      143 |                      DO 4 RIGHT=L SIDE,R SIDE
  6    50    104      144 |                         NEW=CUBE
  6    50    105      145 |                         SHIFT=UP+FORW+RIGHT
  6    50    106      146 |                         IF (SHIFT .GT. 0) NEW=ISLL(NEW,SHIFT)
  6    52    108      147 |                         IF (IAND(SHAPE4,NEW) .EQ. 0) THEN
  7    53    109      148 |                            KIND=0
  7    53    110      149 |                               IF (IAND(VERTEX,NEW) .GT. 0) KIND=KIND+1
  7    55    112      150 |                               IF (IAND(EDGE,NEW) .GT. 0) KIND=KIND+1
  7    57    114      151 |                               IF (IAND(FACE LR,NEW) .GT. 0) KIND=KIND+3
  7    59    116      152 |                               IF (IAND(FACE TB,NEW) .GT. 0) KIND=KIND+3
  7    61    118      153 |                               IF (IAND(FACE FR,NEW) .GT. 0) KIND=KIND+3
  7    63    120      154 |                               IF (IAND(CENTER,NEW) .GT. 0) KIND=KIND-3
  7    65    122      155 |                               IF (KIND .EQ. TYPE(LIST)) THEN
  8    66    123      156 |                                  A(INDEX)=NEW
  8    66    124      157 |                                  INDEX=INDEX+1
  8    66    125      158 |                               END IF
  7    66    126      159 |                         END IF
  6    68    127      160 |    4                 CONTINUE
  5    68    128      161 |    5              CONTINUE
  4    69    129      162 |    6           CONTINUE
  3    70    130      163 |    7        CONTINUE
  2    71    131      164 |             TAIL=INDEX-1
  2    71    132      165 |             CALL ORDNE(A,PIECE,HEAD,TAIL)
  2    71    133      166 |             HEAD=TAIL+1
  2    71    134      167 |             T(LIST)=TAIL
  2    71    135      168 |             LIST=LIST+1
  2    71    136      169 |             H(LIST)=HEAD
  2    71    137      170 |    8     CONTINUE
  1    72    138      171 |    9  CONTINUE
       73    139      172 |       RETURN
       73    140      173 |       END

 ****  SOURCE  LISTING  ****      SIEMENS  FORTRAN  COMPILER   F O R  1     V1.50          PAGE    1

                                                PROGRAM UNIT: MOTION

 DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------

     1/1      1          1 |       INTEGER FUNCTION MOTION(CUBE,LEVEL)
     1/1                 2 |
       1      2          3 |       INTEGER CUBE,LEVEL(2),L1,L2
       1                 4 |
       1      3          5 |       L1=LEVEL(1)
       1      4          6 |       L2=LEVEL(2)
       1      5          7 |       IF (IAND(CUBE,L1) .EQ. CUBE) THEN
  1    2      6          8 |          MOTION=2
  1    2      7          9 |       ELSE IF (IAND(CUBE,L2) .EQ. CUBE) THEN
  1    3      8         10 |          MOTION=1
  1    3      9         11 |       ELSE
  1    4     10         12 |          MOTION=0
  1    4     11         13 |       END IF
       5     12         14 |       RETURN
       5     13         15 |       END

 ****  SOURCE  LISTING  ****      SIEMENS  FORTRAN  COMPILER   F O R  1     V1.50          PAGE    1

                                                PROGRAM UNIT: ORDNE

 DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------

  1/1      1          1 |       SUBROUTINE ORDNE(INPUT,OUTPUT,HEAD,TAIL)
  1/1                 2 |
    1      2          3 |       INTEGER INPUT(TAIL),OUTPUT(TAIL),NEXT(300)
  2/1      3          4 |       INTEGER HEAD,TAIL,FIRST,LAST,POINT,EL,I
  2/1                 5 |
    1      4          6 |       FIRST=HEAD
    1      5          7 |       LAST=HEAD
    2      6          8 |       DO 3 I=HEAD+1,TAIL
    2                 9 |
                     10 |C      Doppelte Elemente werden aussortiert!
    2                11 |
```

```
DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------

 1     3      7        12 |          IF (INPUT(I) .EQ. INPUT(LAST)) GO TO 3
 1     4      9        13 |          IF (INPUT(I) .EQ. INPUT(FIRST)) GO TO 3
 1     4               14 |
                       15 |C     INPUT(I) am Anfang der geordneten Liste vorfügen
 1     4               16 |
 1     5     11        17 |          IF (INPUT(I) .LT. INPUT(FIRST)) THEN
 2     6     12        18 |             NEXT(I)=FIRST
 2     6     13        19 |             FIRST=I
 2     6               20 |
                       21 |C     INPUT(I) am Ende der geordneten Liste anfügen
 2     6               22 |
 2     6     14        23 |          ELSE IF (INPUT(I) .GT. INPUT(LAST)) THEN
 2     7     15        24 |             NEXT(LAST)=I
 2     7     16        25 |             LAST=I
 2     7               26 |
                       27 |C     Suchen in der geordneten Liste und ...
 2     7               28 |
 2     7     17        29 |          ELSE
 2     8     18        30 |             EL=FIRST
 2     9     19        31 |    1        POINT=EL
 2     9     20        32 |             EL=NEXT(EL)
 2     9     21        33 |             IF (INPUT(EL) - INPUT(I)) 1,3,2
 2     9               34 |
                       35 |C     ... INPUT(I) an der richtigen Stelle einfügen
 2     9               36 |
 2    10     22        37 |    2        NEXT(I)=EL
 2    10     23        38 |             NEXT(POINT)=I
 2    10     24        39 |          END IF
 1    12     25        40 |    3 CONTINUE
      12     26        41 |      TAIL=HEAD
      12     27        42 |      EL=FIRST
      12               43 |
                       44 |C     INPUT geordnet abspeichern in OUTPUT
      12               45 |
      13     28        46 |    4 OUTPUT(TAIL)=INPUT(EL)
      13     29        47 |      IF (EL .EQ. LAST) RETURN
      15     31        48 |      TAIL=TAIL+1
      15     32        49 |      EL=NEXT(EL)
      15     33        50 |      GO TO 4
      15     34        51 |      END

****  SOURCE  LISTING  ****      SIEMENS  FORTRAN  COMPILER   F O R  1     V1.50          PAGE    1

                                                PROGRAM UNIT: PUZZLE

DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------

    1/1      1         1 |      SUBROUTINE PUZZLE(PIECE,HEAD,TAIL,NR,RESULT,COUNT,MAX)
    1/1               2 |
      1      2         3 |      INTEGER PIECE(960),HEAD(23),TAIL(22),NR(2:7),RESULT(240,7),COUNT
      1      3         4 |      INTEGER MAX,ASSEMB,LE,BRANCH,MARK,DIFF,NEXT,INDEX,I,LIST,POINTR(7)
      1               5 |
                      6 |C     !! This subroutine uses the functions:
                      7 |C     !! IAND  (=logical AND)
                      8 |C     !! IOR   (=logical OR)
                      9 |C     !! IXOR  (=logical EXCLUSIVE OR)
      1              10 |
      1      4       11 |      BRANCH=3
      1      5       12 |      DIFF=5
      1      6       13 |      MARK=8
      1      7       14 |      POINTR(2)=HEAD(2)-1
      1      8       15 |      ASSEMB=PIECE(POINTR(2))
      1              16 |
                     17 |C     Move the POINTER !
      1              18 |
      2      9       19 |    1 IF (BRANCH .EQ. 3  .AND.  POINTR(2) .EQ. TAIL(2)) RETURN
      4     11       20 |      BRANCH=BRANCH-1
      4     12       21 |      DIFF=DIFF+1
      4     13       22 |      MARK=MARK-DIFF
      4     14       23 |      LE=PIECE(POINTR(BRANCH))
      4     15       24 |      ASSEMB=IXOR(ASSEMB,LE)
      4     16       25 |      IF (POINTR(BRANCH) .EQ. TAIL(MARK)) GO TO 1
      5     18       26 |      POINTR(BRANCH)=POINTR(BRANCH)+1
      5     19       27 |      LE=PIECE(POINTR(BRANCH))
      5     20       28 |      ASSEMB=IOR(ASSEMB,LE)
      5              29 |
                     30 |C     Sort out all pieces there is no place for !
      5              31 |
      6     21       32 |    2 BRANCH=BRANCH+1
      6     22       33 |      MARK=MARK+DIFF
      6     23       34 |      DIFF=DIFF-1
      6     24       35 |      NEXT=HEAD(MARK)
      6     25       36 |      I=MARK
      7     26       37 |      DO 4 LIST=MARK-DIFF,MARK-1
 1    9     27       38 |         DO 3 INDEX=HEAD(LIST),TAIL(LIST)
 2   10     28       39 |            LE=PIECE(INDEX)
 2   10     29       40 |            IF (IAND(ASSEMB,LE) .EQ. 0) THEN
 3   11     30       41 |               PIECE(NEXT)=LE
 3   11     31       42 |               NEXT=NEXT+1
```

```
DO/IF SEG   STMT  I   LINE        SOURCE-TEXT
-------------------------------------------------------------------------------
3    11     32        43 |            END IF
2    13     33        44 |    3     CONTINUE
1    13     34        45 |          IF(NEXT .EQ. HEAD(I)) GO TO 1
1    14     36        46 |          TAIL(I)=NEXT-1
1    14     37        47 |          I=I+1
1    14     38        48 |          HEAD(I)=NEXT
1    14     39        49 |    4 CONTINUE
1    14               50 |
                      51 |C     Store the solution !
1    14               52 |
     15     40        53 |       IF (BRANCH .EQ. 7) THEN
1    16     41        54 |          COUNT=COUNT+1
1    16     42        55 |          IF (MAX .LT. NEXT) MAX=NEXT
1    19     44        56 |          DO 5 LIST=2,6
2    20     45        57 |             RESULT(COUNT,NR(LIST))=PIECE(POINTR(LIST))
2    20     46        58 |    5     CONTINUE
1    21     47        59 |          RESULT(COUNT,NR(7))=PIECE(NEXT-1)
1    21     48        60 |          GO TO 1
1    21     49        61 |       END IF
1    21               62 |
                      63 |C     Set the POINTER the first time in the BRANCH !
1    21               64 |
     22     50        65 |       POINTR(BRANCH)=HEAD(MARK)
     22     51        66 |       LE=PIECE(POINTR(BRANCH))
     22     52        67 |       ASSEMB=IOR(ASSEMB,LE)
     22     53        68 |       GO TO 2
     22               69 |
     22     54        70 |       END
```

References

[1] E. Balas and M.W. Padberg, Set partitioning—A survey, SIAM Rev. 18 (1976) 710–760. Reprinted as Chapter 7 in: N. Christofides, A. Mingozzi, P. Toth and C. Sandi, eds., Combinatorial Optimization (Wiley, New York, 1979) 151–210.

[2] E.R. Berlekamp, J.H. Conway and Richard K. Guy, Winning Ways for your Mathematical Plays, Volume 2: Games in Particular (Academic Press, New York, 1982) 735–739, 801–804.

[3] P. van Delft and Jack Botermans, Creative Puzzles of the World (Plenary Publications International (Europe) b.v., Amstelveen, The Netherlands, 1980).

[4] M. Gardner, The 2nd Scientific American Book of Mathematical Puzzles & Diversions (Simon and Schuster, New York, 1961).

[5] M. Gardner, Mathematical games: pleasurable problems with polycubes, and the winning strategy for Slither, Sci. Amer. 227 (3) (1972) 176–182.

[6] S.W. Golomb, Polyominoes (Charles Scribner's Sons, New York, 1965).