Fundamentals of Artificial Intelligence: Constraint Processing Assignment

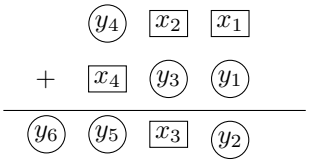
Shiva Hosseini Tehrani

Constraint Programming is widely used in operations research. It is finding a feasible solution rather than optimization and focuses on the constraints and variable domains rather than the objective function.
constraint programming exploit constraints during tree search
1. Use propagation algorithms for constraints
2. Employ branching algorithm
3. Execute exploration algorithm

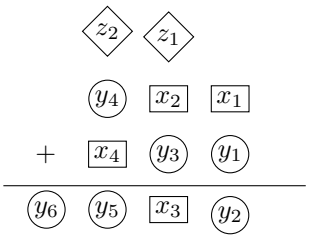
1 Summation Problem

Given a set of even number represented by squares and given a set of odd number represented by circles in the range of $\{1 \dots 6\}$. The problem refers to assigning numbers to the circles and squares such that:
1. The summation of most right digit of the first number which is even and the most right digit of the second number which is an odd number should be odd.
2. The summation of the middle digit of the first number which is even and the middle digit of the second number which is odd and the possible carry should be an even number.
3. The summation of the most left digit of the first number which is odd and the most left digit of the second number which is even and the possible carry should be odd.

We have specified each digit with a variable, as it has shown in the figure below:



To be able to model the problem, we have to take into account the carry digits. These digits are represented by two extra variables z_1 and z_2 as shown in the figure below:



Summation problem can be modeled as a constraint satisfaction problem (CSP), with the following properties (the contents of the field 'submitted variables & constraints' is shown in figure 4 in section 4):

variables:
Even numbers : x_1, x_2, x_3, x_4
Odd numbers : $y_1, y_2, y_3, y_4, y_5, y_6$
Carries : z_1, z_2

domains of variables:
For $x_1 \dots x_4$ the domain is: {2, 4, 6}
For $y_1 \dots y_6$ the domain is: {1, 3, 5}
For z_1, z_2 the domain is: {0,1}

constraints:

$$x_1 + y_1 = y_2 + 10 * z_1 \tag{1}$$
$$x_2 + y_3 = x_3 + 10 * z_2 \tag{2}$$
$$x_4 + y_4 = y_5 + 10 * y_6 \tag{3}$$

Constraints 1, 2 and 3 express the basic rule of summation of numbers: 'Summing two digits gives a digit and possibly a carry digit, if the result of summation has two digits'. A solution is an assignment of a value in domain to each variable such that every constraint satisfied. The software finds 3 solutions for this problem which are shown in figure 1.

2 Stacking squares

In stacking squares the problem is locating a 5×5, 4×4, 3×3, 2×2 in a 9×7 rectangle in a way that non of the squares overlap, squares can not be rotated and only the integer coordinates should be used in locating the corners of the squares.

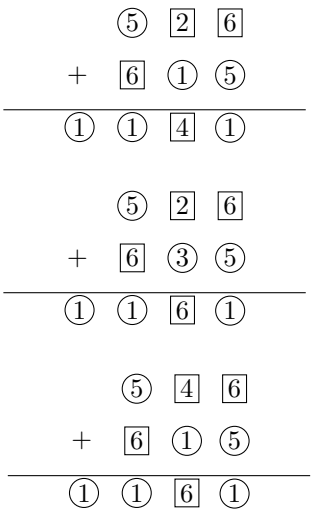


Figure 1: Solutions found by the software for problem 1

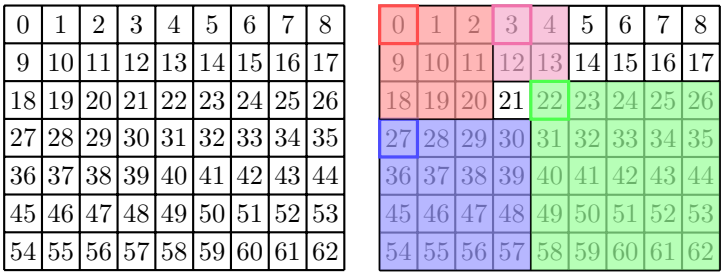


Figure 2: Stacking 4 small rectangles in a large one. We can specify the location of each rectangle using the cell on its upper left corner.

To model the problem, we number the cells in the large rectangle from 0 to 62. The location of each small rectangle can be specified through the location of the cell on its top left. We also define a variable for each cell to denote by which rectangle it is covered (or that it is not covered by any). The problem can be modeled as follows (the contents of the field 'submitted variables & constraints is shown in figure 5 in section 4):

variables:

column and row of upper-left corner of 2×2 rectangle	$S2_c, S2_r$	
column and row of upper-left corner of 3×3 rectangle	$S3_c, S3_r$	
column and row of upper-left corner of 4×4 rectangle	$S4_c, S5_r$	
column and row of upper-left corner of 5×5 rectangle	$S5_c, S5_r$	
column of cell i in the board	$col(i)$	$0 \leq i \leq 62$
row of cell i in the board	$row(i)$	$0 \leq i \leq 62$
the rectangle covering cell i in the board	$Z(i)$	$0 \leq i \leq 62$

$S2_c$ and $S2_r$ are indicating the column and row of top left corner of 2×2 square, $S3_c$ and $S3_r$ are indicating the column and row of top left corner of 3×3 square and so on. $col(i)$ and $row(i)$ represent the column and row of the cell i respectively. $Z(i)$ denotes how cell i is covered. If cell i is covered by 5×5 square, the value of $Z(i)$ will be 5 and so on. If cell i is not covered by any sqaure, $Z(i)$ will be equal to one.

domains of variables:

$$
\begin{array}{llll}
S2_c \in \{0 \dots 7\}, & S2_r \in \{0 \dots 5\} & & \\
S3_c \in \{0 \dots 6\}, & S3_r \in \{0 \dots 4\} & & \\
S2_c \in \{0 \dots 5\}, & S4_r \in \{0 \dots 3\} & & \\
S4_c \in \{0 \dots 4\}, & S5_r \in \{0 \dots 2\} & & \\
col(i) & \in \{0 \dots 8\} & 0 \leq i \leq 62 & \\
row(i) & \in \{0 \dots 6\} & 0 \leq i \leq 62 & \\
Z(i) & \in \{1 \dots 5\} & 0 \leq i \leq 62 &
\end{array}
$$

The first domain indicates that the top left corner of the 2×2 square could be placed in columns $\{0 \dots 7\}$ and rows $\{0 \dots 5\}$ (if it is placed in column 8 or row 6 the square will be located outside of the 9×7 rectangle). The second, third and forth domains have the same view in different ranges for 3×3, 4×4 and 5×5 squares, respectively. The fifth and sixth domains are indicating the range of column and row of each cell in 9×7 rectangle, which can be obtained by the following formula:

$$col(i) = i \mod 9 \quad 0 \leq i \leq 62 \tag{4}$$

$$row(i) = \lfloor \frac{i}{9} \rfloor \quad 0 \leq i \leq 62 \tag{5}$$

constraints:

$$Z(i) = 2 \Leftrightarrow col(i) - 1 \leq S2_c \leq col(i) \wedge \quad (6)$$
$$row(i) - 1 \leq S2_c \leq row(i)$$

$$Z(i) = 3 \Leftrightarrow col(i) - 2 \leq S3_c \leq col(i) \wedge \quad (7)$$
$$row(i) - 2 \leq S3_c \leq row(i)$$

$$Z(i) = 4 \Leftrightarrow col(i) - 3 \leq S4_c \leq col(i) \wedge \quad (8)$$
$$row(i) - 3 \leq S4_c \leq row(i)$$

$$Z(i) = 5 \Leftrightarrow col(i) - 4 \leq S5_c \leq col(i) \wedge \quad (9)$$
$$row(i) - 4 \leq S5_c \leq row(i)$$

$$col(i) = i \mod 9 \quad 0 \leq i \leq 62 \quad (10)$$
$$row(i) = i/9 \quad 0 \leq i \leq 62 \quad (11)$$

Constraint 6 indicates the fact that cell i can be covered by 2×2 square if and only if the topleft corner of that square can be placed in rows $\{row(i) - 1 \ldots row(i)\}$ and columns $\{col(i)-1 \ldots col(i)\}$. For instance in figure 2, $Z(20) = 2$ if and only if $S2C \in \{1, 2\}$ and $S2R \in \{1, 2\}$. The rest constraint 7-9 have the same view in different ranges for 3×3, 4×4 and 5×5 squares, respectively. Constraints 10 and 11 represent formulas 4 and 5.

The software finds 68 solutions for this problem. These solutions are shown in appendix.

3 Docked ships

0	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

0	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

Figure 3: Assigning areas and times to ships can be converted to the problem of stacking rectangles.

This problem can be converted to the problem of stacking squares. Taking the horizontal line as dock area and the vertical line as time, we will have a 4×7 square for the 4 hours that the 7 areas of dock are available. Similarly, we can convert the area and time requirement of each ship into a rectangle. Finally,

the problem is to fit these four rectangles in the large 4×7 rectangle. The problem can be modeled as follows (the contents of the field 'submitted variables & constraints is shown in figure 6 in section 4):

variables:

column and row of upper-left corner of rectangle for ship 1	$S1_c, S1_r$	
column and row of upper-left corner of rectangle for ship 2	$S2_c, S2_r$	
column and row of upper-left corner of rectangle for ship 3	$S3_c, S3_r$	
column and row of upper-left corner of rectangle for ship 4	$S4_c, S4_r$	
column of cell i in the board	$col(i)$	$0 \leq i \leq 27$
row of cell i in the board	$row(i)$	$0 \leq i \leq 27$
the rectangle covering cell i in the board	$Z(i)$	$0 \leq i \leq 27$

domains of variables:

$$
\begin{array}{llll}
S1_c \in \{0 \ldots 3\}, & S1_r \in \{0, 1\} & & \\
S2_c \in \{0 \ldots 5\}, & S2_r \in \{0 \ldots 2\} & & \\
S3_c \in \{0 \ldots 4\}, & S3_r \in \{0 \ldots 3\} & & \\
S4_c \in \{0 \ldots 2\}, & S4_r \in \{0 \ldots 3\} & & \\
col(i) & \in \{0 \ldots 6\} & 0 \leq i \leq 27 & \\
row(i) & \in \{0 \ldots 3\} & 0 \leq i \leq 27 & \\
Z(i) & \in \{0 \ldots 4\} & 0 \leq i \leq 27 &
\end{array}
$$

constraints:

$$Z(i) = 1 \Leftrightarrow col(i) - 3 \leq S1_c \leq col(i) \wedge \quad (12)$$
$$row(i) - 2 \leq S1_r \leq row(i)$$

$$Z(i) = 2 \Leftrightarrow col(i) - 1 \leq S2_c \leq col(i) \wedge \quad (13)$$
$$row(i) - 1 \leq S2_r \leq row(i)$$

$$Z(i) = 3 \Leftrightarrow col(i) - 2 \leq S3_c \leq col(i) \wedge \quad (14)$$
$$S3_r = row(i)$$

$$Z(i) = 4 \Leftrightarrow col(i) - 4 \leq S4_c \leq col(i) \wedge \quad (15)$$
$$S4_r = row(i)$$

$$col(i) = i \mod 7 \quad 0 \leq i \leq 27 \quad (16)$$
$$row(i) = i/7 \quad 0 \leq i \leq 27 \quad (17)$$

Variables and constraints have the same meaning as in the previous problem. The only difference is that in this problem, $Z(i) = j, 1 \leq j \leq 4$ means that cell i is assigned to ship j and $Z(i) = 0$ means that cell i is not covered.

The software finds 56 solutions for this problem. The solutions are shown in appendix.

Comparing Heuristics

In many cases, using *first-fail* heuristic for this problem leads to several backtracks, while the number of backtracks using *leftmost* heuristic is small. The reason can be that when we use *leftmost* heuristic, the value of S variables is first selected, and then the value of Z variables is derived, but when we use *first-fail*, the value of a Z variable may be assigned first, and later it makes it impossible to find values for other variables.

4 Submitted Variables and Constraints

```
Name: X(i), i=1..4, domain: 2..6
Name: Y(i), i=1..6, domain: 1..5
Name: Z(i), i=1..2, domain: 0..1

Constraint: X(i) mod 2 = 0,
range: (i >= 1 /\ i =< 4 )

Constraint: Y(i) mod 2 = 1,
range: (i >= 1 /\ i =< 6)

Constraint: X(i) + Y(i) = Y(j) + 10 * Z(i),
range: i = 1 /\ j = 2

Constraint: Z(i) + X(j) + Y(k) = X(k) + 10*Z(j),
range: i = 1 /\ j = 2 /\ k = 3

Constraint: Z(i) + Y(j) + X(j)  =Y(k) + 10*Y(l),
range: i = 2 /\ j = 4 /\ k = 5 /\ l = 6
```

Figure 4: *Submitted variables and constraints* for problem 1

```
Name: S2C, domain: 0..7
Name: S2R, domain: 0..5
Name: S3C, domain: 0..6
Name: S3R, domain: 0..4
Name: S4C, domain: 0..5
Name: S4R, domain: 0..3
Name: S5C, domain: 0..4
Name: S5R, domain: 0..2
Name: Z, domain: 1..5
Name: Col, domain: 0..8
Name: Row, domain: 0..6

Constraint: Z(i) = 2 <=>
(Col(i) - 1 =< S2C /\ S2C =< Col(i)
/\ Row(i) - 1 =< S2R /\ S2R =< Row(i)) ,
range: 0 =< i /\ i<=62

Constraint: Z(i) = 3 <=>
(Col(i) - 2 =< S3C /\ S3C =< Col(i)
/\ Row(i) - 2 =< S3R /\ S3R =< Row(i)) ,
range: 0 =< i /\ i<=62

Constraint: Z(i) = 4 <=>
(Col(i) - 3 =< S4C /\ S4C =< Col(i)
/\ Row(i) - 3 =< S4R /\ S4R =< Row(i)) ,
range: 0 =< i /\ i<=62

Constraint: Z(i) = 5 <=>
(Col(i) - 4 =< S5C /\ S5C =< Col(i)
/\ Row(i) - 4 =< S5R /\ S5R =< Row(i)) ,
range: 0 =< i /\ i<=62

Constraint: Col (i) = i mod 9 , range: 0 =< i /\ i<=62
Constraint: Row(i) = ( i / 9 ), range: 0 =< i /\ i<=62
```

Figure 5: *Submitted variables and constraints* for problem 2

```
Name: S1C, domain: 0..3
Name: S1R, domain: 0..1
Name: S2C, domain: 0..5
Name: S2R, domain: 0..2
Name: S3C, domain: 0..4
Name: S3R, domain: 0..3
Name: S4C, domain: 0..2
Name: S4R, domain: 0..3
Name: Col(i), i=0..27, domain: 0..6
Name: Row(i), i=0..27, domain: 0..3
Name: Z(i), i=0..27, domain: 0..4

Constraint: Col(i) = i mod 7, range: 0 =< i =< 27
Constraint: Row(i) = i / 7, range: 0 =< i =< 27

Constraint: Z(i) = 1 <=>
(Col(i) - 3 =< S1C /\ S1C =<  Col(i)
/\ Row(i) -2 =< S1R /\ S1R=< Row(i)),
range: 0 =< i /\ i =< 27

Constraint: Z(i) = 2 <=>
(Col(i) - 1 =< S2C /\ S2C =<  Col(i)
/\ Row(i) -1 =< S2R /\ S2R=< Row(i)),
range: 0 =< i /\ i =< 27

Constraint: Z(i) = 3 <=>
(Col(i) - 2 =< S3C /\ S3C =<  Col(i)
/\ S3R= Row(i)),
range: 0 =< i /\ i =< 27

Constraint: Z(i) = 4 <=>
(Col(i) - 4 =< S4C /\ S4C =<  Col(i)
/\ Row(i) = S4R),
range: 0 =< i /\ i =< 27
```

Figure 6: *Submitted variables and constraints* for problem 3