The 3-Jug Problem (jug)

—- Run LPL Code , HTML Document –

Problem: There are three jugs with capacities of 8, 5, and 3 liters. Initially the 8-liter jug is full of water, whereas the others are empty. Find a sequence for pouring the water from one jug to another such that the end result is to have 4 liters in the 8-liter jug and the other 4 liters in the 5-liter jug. When pouring the water from a jug A into another jug B, either jug A must be emptied or B must be filled, see Figure 1.

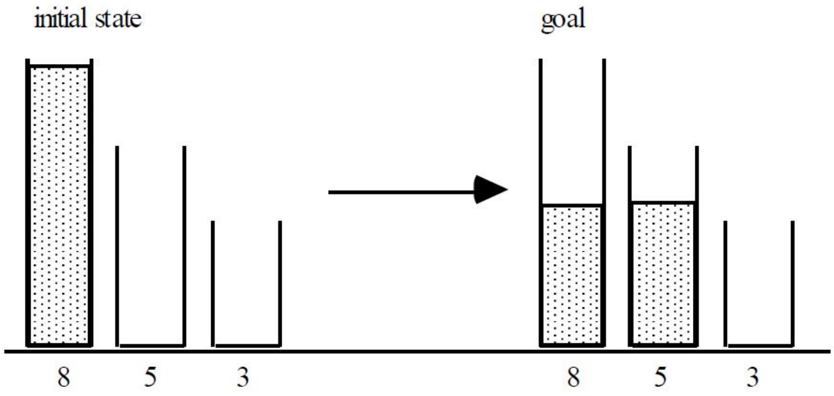

Figure 1: 3-Jug Problem

Modeling Steps

The problem can be formulated as a number of *states* and *transitions* between the states. A state is a particular filling of the jugs, for instance, "the 8-liter jug contains 8 liters of water, the 5-liter jug contains nothing, and the 3-liter jug contains nothing." Each state can be represented by a triple of numbers: (x, y, z), where x is the content of the 8-liter jug, y is the content of the 5-liter jug, and z is the content of the 3-liter jug. So, $(8, 0, 0)$ is the example just used before. We want to reach the state $(4, 4, 0)$. The capacities of the jug can also be represented by a triple: $(8, 5, 3)$. The first step is to enumerated all states. There are 216 potential states, namely, all states (x, y, z) with $0 \leq x \leq 8$, $0 \leq y \leq 5$, and $0 \leq z \leq 3$, which is $9 \cdot 6 \cdot 4 = 216$. However, only states where $x + y + z = 8$ are allowed (no water should be added or removed). Furthermore, at least one jug must be full or empty – following the rules of pouring. This condition can be represented as a Boolean expression as follows:

$$x = 0 \lor x = 8 \lor y = 0 \lor y = 5 \lor z = 0 \lor z = 3$$

There are 16 remaining possible states, they are:

$$\{(0, 5, 3), (1, 4, 3), (1, 5, 2), (2, 3, 3), (2, 5, 1), (3, 2, 3), (3, 5, 0), (4, 1, 3), (4, 4, 0), (5, 0, 3), (5, 3, 0), (6, 0, 2), (6, 2, 0), (7, 0, 1), (7, 1, 0), (8, 0, 0)\}$$

The basic operation is to pour water from one jug A to another jug B in a way that either A is emptied or B is filled. We are looking for the shortest sequence of operations that reaches the state $(4, 4, 0)$ starting with state $(8, 0, 0)$. Such a basic operation is called a *transition from one state to another*. The result is a (directed) graph. The problem now is reduced to find the shortest (direct) path in this graph from state $(8, 0, 0)$ to the state $(4, 4, 0)$. The resulting graph and the shortest path in red is given in Figure 2.

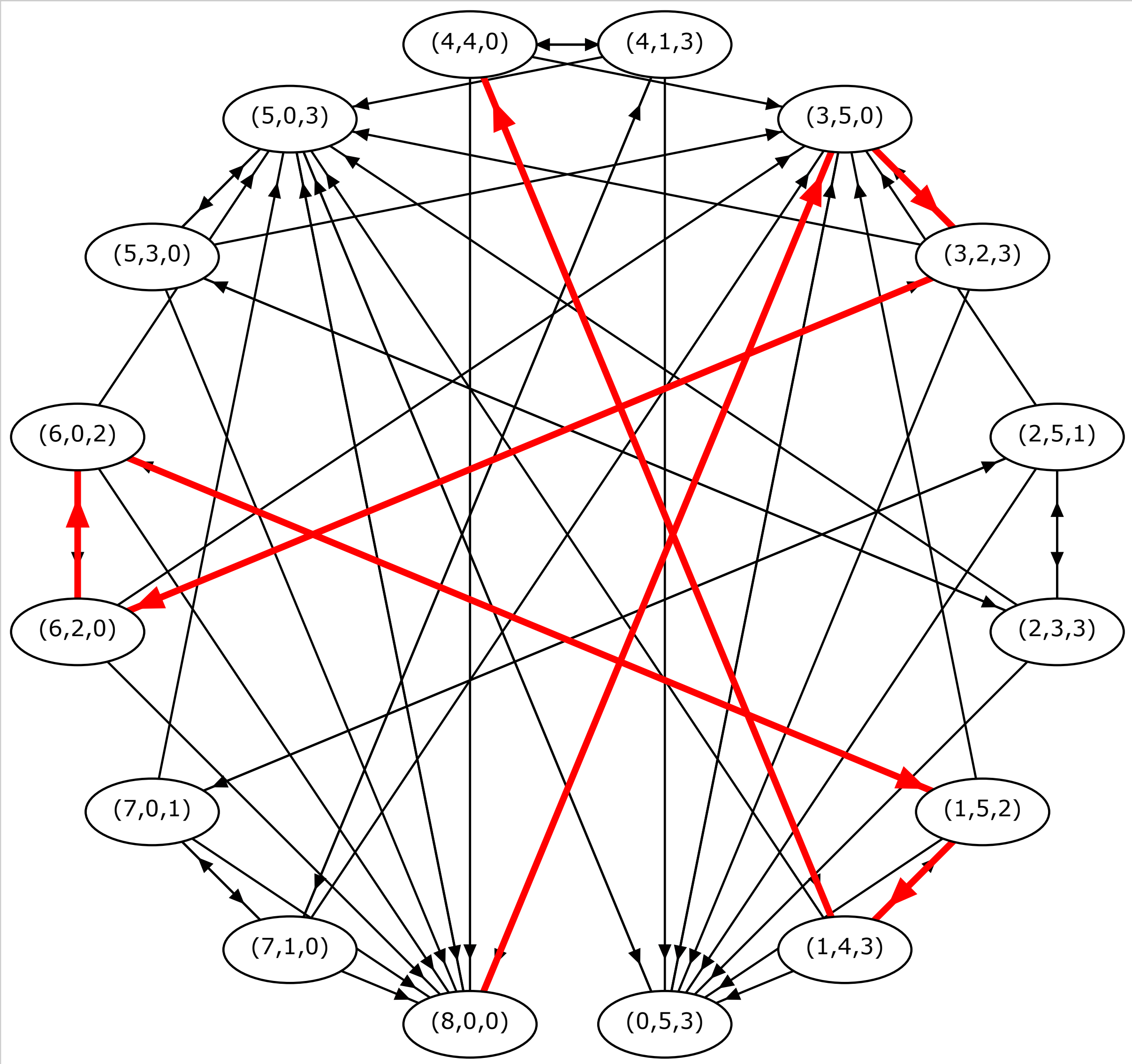

Figure 2: Solution of the 3-Jug Problem

Further Comments: The model `EnumerateStates` returns all possible states and S_i contains all state names as strings. The most challenging is to calculated the transsitions $e_{i,j}$ between state i and j: Let k be a jug then $k\%N+1$[1] is the next jug and $(k+1)\%N+1$ is the next but one. A valid pouring from state i to state j for jug $k\%N+1$ to jug $(k+1)\%N+1$ is defined as follows: (1) not touching the jug k, (that is $x_{k,i} = x_{k,j}$) *and* (2) either filling the jug $(k+1)\%N+1$ of state j (that is, $x_{(k+1)\%N+1,j} = C_{(k+1)\%N+1]}$) or emptying the jug $k\%N+1$ of state j (that is, $x_{k\%N+1,j} = 0$). Totally six of these conditions must hold, because there are six pouring possibilities with three jugs. The conditions are implemented in the `e{i,j}:=...` statement.

Calling to function `Graph.SPath` returns the shortest path.

Listing 1: The Complete Model implemented in LPL [2]

```
model jug "The 3-Jug Problem";
  set k        "Set of jugs";
    i,j        "The set of states";
```

[1]Note: % is the modulo operator.

```
    e{i,j}     "Transition links";
    p{i,j}     "shortest path";
  parameter N "Number of jugs";
    T          "Total liquid";
    C{k}       "capacities of the jugs";
    x{k,i}     "the states";
  string S{i} "State name";
    InitState "Starting state";
    GoalState "Arrivng State";
  ;EnumerateStates;
  S{i}:='('&x[1,i]&','&x[2,i]&','&x[3,i]&')';
  e{i,j}:= or{k} (
    x[k,i]=x[k,j] and (x[k%N+1,j]=C[k%N+1] or x[(k+1)%N+1,j]=0) or
    x[k,i]=x[k,j] and (x[k%N+1,j]=0 or x[(k+1)%N+1,j]=C[(k+1)%N+1]));
  parameter s:=argmax{i} (S=InitState) "from";
            t:=argmax{i} (S=GoalState) "to";
  if Graph.SPath(e,p,s,t)>=99999999 then Write('No␣path␣exists\n'); end
      ;
  --DrawGraph;
  DrawJug(8,0,0, 0,0);
  DrawJug(4,4,0, 25,0);
  Draw.Arrow(16,4,23,4,-2,0,3);
  Draw.Text('Abstraction',15,-3.5,16,0,3,1.3);
  parameter SPdraw{i}:=[16,7,6,13,12,3,2,9]; n;
  parameter SPdraw1{i}:=[16,9];
  {i|SPdraw1} (n:=n+1, Draw.Ellipse(S[SPdraw1],if(n>4,(n-4)*10-5,n
      *25-17),if(n>4,-10,-6),3,1,1,0));
  Draw.Arrow(16,-6,23,-6,-2,0,3);
/*  {i|SPdraw} (n:=n+1, Draw.Ellipse(S[SPdraw],if(n>4,(n-4)*10-5,n
    *10-5),if(n>4,-10,-6),3,1,1,0));
  {p in 1..3} Draw.Arrow(p*10-2,-6,p*10+2,-6);
  {p in 1..3} Draw.Arrow(p*10-2,-10,p*10+2,-10);
  Draw.Line(40-5,-7,40-5,-8);
  Draw.Line(40-5,-8,5,-8);
  Draw.Arrow(5,-8,5,-9);
*/
  model data;
    T:=8;
    InitState:='(8,0,0)'; //sum must be T
    GoalState:='(4,4,0)'; //sum must be T
    N:=3;
    k:=1..N;
    C{k}:=[8,5,3];
  end;
  model EnumerateStates;
    {a in 0..C[1],b in 0..C[2], c in 0..C[3]}
      if(a+b+c=T and (a=0 or a=C[1] or b=0 or b=C[2] or c=0 or c=C[3]),
        (Addm(i,#i+1), x[1,#i]:=a,x[2,#i]:=b,x[3,#i]:=c)
      );
  end
  model DrawGraph;
    parameter
      PI:=3.14159;
      xa{i}:=15*Sin(PI/#i+2*PI*(i-1)/#i);
      ya{i}:=15*Cos(PI/#i+2*PI*(i-1)/#i);
    Draw.Scale(15,15);
    Draw.DefFont('Verdana',10);
    {e[i,j]} Draw.Arrow(xa[i],ya[i],xa[j],ya[j],2);
```

```
    --{p[i,j]} Draw.Arrow(xa[j],ya[j],xa[i],ya[i],2,3,3);
    {i}Draw.Ellipse(S,xa,ya,2,1,1,0);
  end
  model DrawJug(integer a;b;c; x;y);
    Draw.Scale(10,-15);
    Draw.DefFont('Verdana',12);
    model oneJug(integer h;a; x;y);
      Draw.Ellipse(x+2,y,2,.5,if(a>0,5,1),0);
      Draw.Rect(x,y,4,1,1,1);
      if a>0 then
        Draw.Rect(x,y,4,y+a,5,5);
        Draw.Ellipse(x+2,a,2,.5,5,1);
      end;
      Draw.Ellipse(x+2,h,2,.5,1,0));
      Draw.Line(x,h,x,0);
      Draw.Line(x+4,h,x+4,0);
      Draw.Text(h&'L',x+1.5,-1.3);
      Draw.Text(a&'L',x+1.5,1);
    end;
    oneJug(8,a,x,y);
    oneJug(5,b,x+5,y);
    oneJug(3,c,x+10,y);
  end
end
```

Questions

1. Vary the problem: Try, for example:

   ```
   T:=9; InitState:='(8,1,0)'; GoalState:='(2,5,2)';
   ```

   Or try this:

   ```
   T:=10; InitState:='(10,0,0)'; GoalState:='(3,4,3)';
   C{k}:=[10 5 3];
   ```

2. Try also this : (what happens?)

   ```
   T:=12; InitState:='(10,2,0)'; GoalState:='(5,6,1)';
   C{k}:=[12 6 3];
   ```

3. Another shorter formulation of the jug problem is: jugA[2].

Answers

1. Just modify the model correspondingly and run again. The nice example to introduce your children to graph

2. No path exists between these two states.

[2]https://lpl.matmod.ch/lpl/Solver.jsp?name=/jugA

References

[1] MatMod. Homepage for Learning Mathematical Modeling : https://matmod.ch.

[2] Hürlimann T. Reference Manual for the LPL Modeling Language, most recent version. https://matmod.ch/lpl/doc/manual.pdf.