U-SIT And Think News Letter - 54

Updates and Commentary

Unified **S**tructured **I**nventive **T**hinking is a problem-solving methodology for creating unconventional perspectives of a problem, and discovering innovative solution concepts, when conventional methodology has waned.

Dear Readers:

. This mini-lecture continues the discussion of heuristics with an example of “symmetry”. How would you apply this heuristic?

. Frederic Mikusek’s variation of the 20,000 lights problem is solved and our solutions are given.

3. Mini USIT Lecture – 54
5. Heuristics for Solving Technical Problems

Using the heuristic “SYMMETRY”

The symmetry heuristic suggests looking for symmetry in a problem. If it exists take it to extremes; examine zero symmetry and examine infinite symmetry, whatever those ideas suggest to you. If there is no symmetry, try introducing symmetry. Of these suggestions, introducing symmetry may be the least familiar. Can you think of a situation where you introduced symmetry to solve a problem?

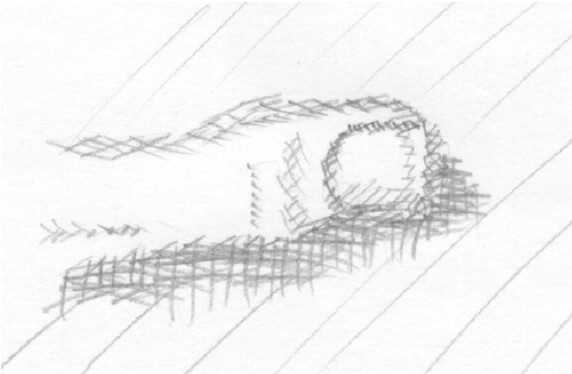
In condensed-matter physics symmetry is a very important idea that is used to characterize and understand crystalline matter. It is also used to solve complex theoretical problems where introducing symmetry often simplifies equations to be solved, e.g., introducing symmetry such as periodicity having a unit cell of a crystal’s lattice as its basic element.

An area of problem solving that I find interesting is art, especially in sketching and painting. In one type of sketching, symmetry is introduced to solve the problem of rendering a three-dimensional impression. In this case the symmetry is not in composition but in technique. Can you think of an example? I have in mind the use of parallel lines to develop degrees of shading by crosshatching.

What intrigues me about crosshatching is its ability to suggest an object without the object being defined by outlines. The result of this technique has always struck me as a bit magical; it causes me to see something that shouldn’t exist since it has no definitive outlines. This is a good example of thinking by the non-language brain hemisphere, the intuitive side of one’s brain. See example on the next page.

In this case symmetry is used to create the object <u>information</u> by the use of parallel lines. The lines can be thought of as objects having attributes of weight, spacing, length, width, curvature, and density (or weight).

Art, in my mind, is another field of problem solving.



****

The 20,000 lights problem.

The 20,000 lights problem was introduced for you to solve as an example of what one can do with the heuristic Simplify! In discussing its solution it was demonstrated that one approach to simplifying the solution process is a graphic technique examining only a portion of the system of lights and persons.

A matrix was constructed having columns labeled “Light Number” and rows labeled “Ordinal Number of Person” (both are ordinal numbers). By marking boxes of the matrix to indicate the new state of a light when its off-on state changes, a graphic image of which lights are on and which are off was created. The resulting image was a triangular array whose hypotenuse (the principal diagonal of the square matrix) showed the final state of each light. The complimentary array has no information in it. Examination of the diagonal revealed useful patterns that could be expressed mathematically and then extrapolated to any number of lights. Two patterns were used to check agreement for the number of on lights in a group of 20,000.

We now have for consideration, an interesting variation of this problem proposed by Frederic Mikusek; if the lights have three states how many remain on? Here’ the original problem and Mikusek’s variation:

> *In a long hall are 20,000 electric lights that are operated each by a pull-chain. Initially all lights are turned off. A person walks through the hall and pulls every chain, thus turning on each light. A second person walks through and pulls every other chain – #2, #4, #6, etc. up to, and including #20,000 – thus turning off those lights. The next person walks through and pulls every third chain – #3, #6, #9, etc. – thus turning off some lights and turning on others. The next person pulls every 4th chain, the next every 5th chain, and so on until the 20,000th person passes through and pulls the chain of #20,000. The question is, after the 20,000th person has pulled the last chain, how many lights remain on?*
>
> Mikusek’s variation: “*Now what if the light changes from off to green, then from green to red, then off again? How many green and red lights remain on?*

My first reaction on reading this variation was to wonder what the relative sizes of the numbers of green, red, and off lights might be? An intuitive answer came to mind – maybe they are equal. However, I had no idea how to justify that answer, so I ignored it and moved on.
Since the original version of the problem was solved by simplification using a small matrix, I started there. Suspecting that a larger matrix would be needed to see any patterns for a three-state system, I used a 26 by 26 matrix (convenient for printing). The following results were found on its diagonal:

Light Number																									
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
g	r	r	-	r	g	r	g	-	g	r	-	r	g	g	r	r	-	r	-	g	g	r	r	-	g

I studied these results for some time but couldn't identify any pattern. I then decided to try a different approach; I wrote a computer program using the programming feature of Mathcad 11. The program (shown below for the interested programmers) and its results follow:

```
st := 20,000 lights

Dgnl(st) := | col ← 2
            | for i ∈ 1..st
            |    | Cell_i ← "G"
            |    | i ← i + 1
            | step ← 2
            | for row ∈ 2..st
            |    | for k ∈ col, col + step..st
            |    |    | Cell_k ← "O"  if Cell_k = "R"
            |    |    | Cell_k ← "R"  if Cell_k = "G"
            |    |    | Cell_k ← "G"  otherwise
            |    |    | k ← k + step
            |    | col ← col + 1
            |    | step ← step + 1
            |    | row ← row + 1
            | Lg_1 ← 0
            | Lg_2 ← 0
            | Lg_3 ← 0
            | for i ∈ 1..st
            |    | Lg_1 ← Lg_1 + 1  if Cell_i = "O"
            |    | Lg_2 ← Lg_2 + 1  if Cell_i = "G"
            |    | Lg_3 ← Lg_3 + 1  otherwise
            |    | i ← i + 1
            | return Cell
```

*

DGNL(st) is a vector of the diagonal values of an st x st matrix.

i = vector cell-index.

Vector's cells initialized to G representing the 1st row of the matrix.

Move to row 2.

k = matrix column.

Change state of cells in column k.

Jump to next k
Inc. matrix column

Inc. step size

Inc. matrix row

Initialize counters for number of off, green, and red lights

Count number of

Off,

Green, and

Red lights.

Increment loop.

Return Cell

Dgnl(st) =

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

$$Ng(st) = \begin{pmatrix} 5.373 \times 10^3 \\ 7.318 \times 10^3 \\ 7.309 \times 10^3 \end{pmatrix} \quad \begin{array}{l} \text{off (0)} \\ \text{green (1)} \\ \text{red (2)} \end{array}$$

5,373 off lights
7,318 green lights
7,309 red lights

Two opportunities for simplification presented themselves: 1) consider only a triangular array above the principal matrix diagonal; 2) Calculate and save only the diagonal elements, thus requiring only a vector.

Frederick Mikusek also solved this problem using a computer program. His is written in Basic and is a bit simpler and easier to read than mine. He writes …

```
green = 0; red = 0; color = 0; n = 20000
Do Until j = n
j = j + 1
i = 0
color = 0
Do Until i = j
i = i + 1
If (j / i) = Int(j / i) Then color = color + 1
'Remark : this means if the man number i will trigger a change for light number j ; j is a multiple of i
If color = 3 Then color = 0
'Remark : replace 3 (0=off ; 1=green ; 2=red)  by 2 ( 0=off ; 1=on) and you find 141 ;-)  [The answer to the original problem.]
Loop
If color = 1 Then green = green + 1
If color = 2 Then red = red + 1
Loop
```

His results are: 7318 green lamps on and 7309 red lamps on and 5373 lamps off. Furthermore, he tested his program for the 2-state case and got the correct results. (I didn't think to do that! ☹)

He also discusses the heuristics he used – the main point of our on-going discourse. Here are his comments with mine italicized in brackets.

Remark : Here are some heuristics used :
"if you have to count something, maths are useful"
"if you can not do it by yourself have it done by somebody else or by a machine" ; *[Yes!]*
"search for a pattern";
"search for a change in a pattern and search what triggers it" ;   *[Root cause – very useful]*
"first think little" (I tried first with a hand filled matrix 37*37 to check the results, unfortunately there were some mistakes because I forgot that 36=6*6 ("symmetry")).
"simplify" (half of the matrix need not be filled);
"local quality" (only some cells of the matrix are useful); *[A big time saver]*
"use colours" (to see patterns);
"check results several times" (until algorithm gives same answers as the hand filled matrix) ; *[Mathematics usually allows independent checks – very useful.]*
"universality" (algorithm should work for any positive number);
"try something new" (I tried visual basic for the first time because I did not see any obvious mathematical analytical solution);
"solve a problem to discover which heuristics you use"(meta-heuristic);
"use your time for fun and learn something new everyday". *[Makes getting up in the morning worthwhile.]*

This is a nice example of thorough introspection to identify heuristics that might otherwise go unnoticed – a practice that reinforces one's facility with the use of heuristics. Notice that his way of using a particular heuristic may differ from your use of the same heuristic. That's fine. Heuristics are personal tools that we use sometimes without even understanding quite how to explain what we are doing.

Frederick ends his email with a request:

"I wish you will give us more heuristics about heuristics in 2006."

I plan to do just that.

*************************** To be continued ********************************

7. Papers and essays

The following materials can be read by clicking on their titles. Links are also available on the USIT website (www.u-sit.net/Publications)

1. "Injecting Creative Thinking Into Product Flow"
2. "Problem Statement"
3. "Metaphorical Observations"

8. Other Interests

1. Have a look at the USIT textbook, "Unified Structured Inventive Thinking – How to Invent", details may be found at the Ntelleck website: www.u-sit.net (*Note*; not at www.ic.net)
2. USIT Resources Visit www.u-sit.net and click on Registration.

Publications	Language	Translators	Available at …
1. Textbook: **Unified Structured Inventive Thinking – How to Invent**	English	Ed Sickafus (author)	www.u-sit.net
2. eBook: **Unified Structured Inventive Thinking – an Overview**	English	Ed Sickafus (author)	www.u-sit.net
	Japanese	Keishi Kawamo, Shigeomi Koshimizu and Toru Nakagawa	www.osaka-gu.ac.jp/php/nakagawa/TRIZ/
	Korean	Yong-Taek Park	www.ktriza.com/www/usit/register_form.htm
"**Pensamiento Inventivo Estructurado Unificado – Una Apreciación Global**"	Spanish	Juan Carlos Nishiyama y Carlos Eduardo Requena	www.u-sit.net
3. eBook "**Heuristics for Solving Technical Problems – Theory, Derivation, Application**" -- **HSTP**	English	Ed Sickafus (author)	www.u-sit.net
"**Heurísticas para Resolver Problemas técnicos – Teoría Deducción Aplicación**"	Spanish	Juan Carlos Nishiyama y Carlos Eduardo Requena	www.u-sit.net
4. **U-SIT and Think Newsletter**	English	Ed Sickafus (Editor)	www.u-sit.net
	Japanese	Toru Nakagawa and Hideaki Kosha	www.osaka-gu.ac.jp/php/nakagawa/TRIZ/
	Korean	Yong-Taek Park	www.ktriza.com.
Mini-lectures from NL_01 through NL_51	Spanish	Juan Carlos Nishiyama y Carlos Eduardo Requena	www.u-sit.net click on Registration

Please send your feedback and suggestions to Ntelleck@u-sit.net and visit www.u-sit.net

To be creative, U-SIT and think.