1st International Conference on Engineering and Technology Development
(ICETD 2012)
Universitas Bandar Lampung
Faculty od Engineering and Faculty of Computer Science
ISSN 2301-6590

Solving Pixel Puzzle Using Rule-Based Techniques and Best First Search

Dina Stefani[#1], Arnold Aribowo[#2], Kie Van IvankySaputra [#3], Samuel Lukas [#3]

[#1]Informatics Engineering Department
[#2] Computer Systems Department
[#3] Applied Mathematics Department
UniversitasPelitaHarapan
UPH Tower, LippoKarawaci, Tangerang, 15811, Indonesia
[1]stefani_dina@hotmail.com
[2]arnold.aribowo@uph.edu
[3]kie.saputra@uph.edu
[4]samuel.lukas@uph.edu

Abstract—**Pixel puzzle is a logic puzzle which consists of a blank grid with clues on the left of every row and on the top of every column. The objective is to paint blocks in each row and column so their length and sequence corresponds to the clues, and there is at least one empty square between adjacent blocks. There are many possible solutions to paint blocks in each row and column. Solving the puzzle manually gives the possibility to fill cells yield erroneously. Therefore an attempt to solve the puzzle with the aid of computer software is performed.**

In this paper, rule–based techniques and best-first search are utilized to solve the puzzle. According to experiments have been conducted, it can be concluded that rule-based techniques and best-first search are able to solve the Pixel Puzzle. The result also indicates that the larger size of pixel puzzle, the longer average time to solve is needed. Moreover, the average time to solve one cell of pixel puzzle depends on the size itself except for the 10 × 10 and 15 × 15 pixels.

Keywords— **Pixel Puzzle, Heuristics, Rule-Based, Best-First Search, Puzzle Solver**

I. Introduction

Solving a puzzle is one of a challenging activity during the leisure time. One of the popular puzzles is Pixel puzzle. Pixel puzzle is also called as Nonogram or Japanase Puzzle. Pixel puzzle is a logic puzzle which consists of a blank grid with clues on the left of every row and on the top of every column. The objective is to paint blocks in each row and column so their length and sequence corresponds to the clues, and there is at least one empty square between adjacent blocks. Usually the result of filled cells forms an image.

Solving the puzzle needs a lot of patience due to the fact that there are many possibilities to paint blocks in each row and column. Solving the puzzle manually also gives the possibility to fill cells erroneously. Therefore an attempt to solve the puzzle with the aid of computer software is needed. Rule–based techniques and best-first search are applied to solve the puzzle. Rule-based techniques which are applied to solve the problem consist of simple boxes, simple spaces, forcing, and contradiction. When rule-based techniques can not solved the problem, the process of finding the solution will be continued by using best-first search. Although some pixel puzzle enable cells to be filled with various color, however, in this paper, cells is filled with black or while color. The maximum size of pixel puzzle is 25x25. To give a more insight about the pixel puzzle, an example of 12x9 pixel puzzle is depicted in the following figure 1:

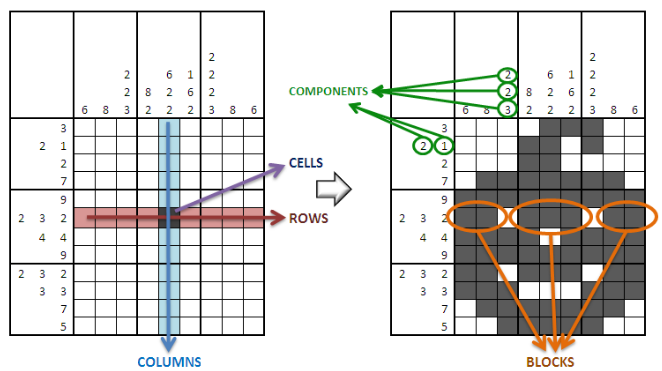


Figure 1. An example of *pixel puzzle*
(left picture is the problem while right picture is the solution)

There are several papers which discuss how to solve the pixel puzzle. In [5], ad-hoc heuristics is implemented. It uses the information in rows, columns, and puzzle's constraints to obtain the solution of the puzzle. In [4], logical rules and depth first search algorithm are implemented. In [1], many pixel puzzle are solved, however some puzzle can not be solved well.

II. PIXEL PUZZLE

ISSN 2301-6590

Nonograms are picture logic puzzles in which cells in a grid have to be coloured or left blank according to numerical clues given at the side and top of the grid in order to reveal a hidden picture. In this puzzle type, the numbers measure how many unbroken lines of filled-in squares there are in any given row or column [3]. For example, a clue of "4 8 3" would mean there are sets of four, eight, and three filled squares, in that order, with at least one blank square between successive groups.

Nonograms are also known by many other names, including Paint by Numbers, Griddlers, Pic-a-Pix, Picross, Shady Puzzles, Pixel Puzzles, Crucipixel, Edel, FigurePic, gameLO, Grafilogika, Hanjie, Illust-Logic, Japanese Crosswords, Japanese Puzzles, KareKarala!, Logic Art, Logic Square, Logicolor, Logik-Puzzles, Logimage, Obrazkilogiczne, Zakódovanéobrázky, Malovanékrížovky, Oekaki Logic, Oekaki-Mate, Paint Logic, ShchorUftor, Gobelini, and Tsunamii. These puzzles are often black and white but can also have some colours.

This paper discusses the black and white Pixel Puzzle. There is no theoretical limit on the size of a pixel puzzle, and they are also not restricted to square layouts. Pixel puzzle was originally invented in 1987 by both Non Ishida, a Japanese graphics editor, and Tetsuya Nishio, a professional Japanese puzzler (although with no connection between them). Pixel puzzle now appear in many newspapers and gaming publications, along with other popular puzzles such as crosswords and Sudoku. Basically, pixel puzzle rules are described as follows:

1) Fills cells in a grid with black colour or left blank according to numerical clues given at the side and top of the grid in order to reveal a hidden picture.
2) There is at least one blank square between successive filled squares.

There is no method to determine precisely the difficulty level of problems in this puzzle. However, there are basically two factors which influence this, namely the size of puzzle and the ratio between the block length and the size. The more pixels, the harder the puzzle will be. In addition to that, the more relatively long blocks there are the easier the puzzle will be. There are several rules which can be employed to solve pixel puzzle, for instance simple boxes, simple spaces, forcing, glue, joining and splitting, mercury, and contradiction [3]. In this paper, glue, joining and splitting, and mercury are not used to solve the problem.

A. *Simple Boxes*

Simple Boxes is basically used to determine as many boxes as possible at the beginning. This method uses conjunctions of possible places for each block of boxes. It is important to note that boxes can be placed in cells only when the same block overlaps. The following figure illustrates the simple boxes rule:

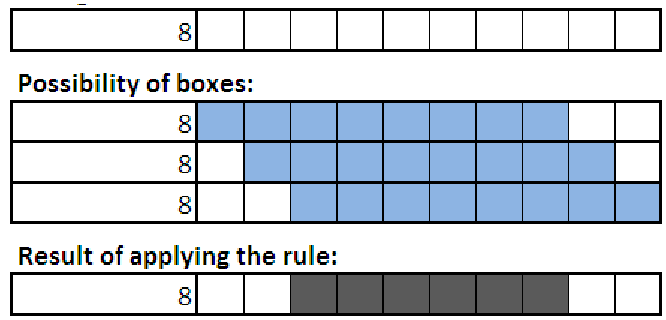


Figure 2. Simple Boxes

B. *Simple Spaces*

This method consists of determining spaces by searching for cells that are out of range of any possible blocks of boxes. The simple spaces rule can be depicted in the following figure:

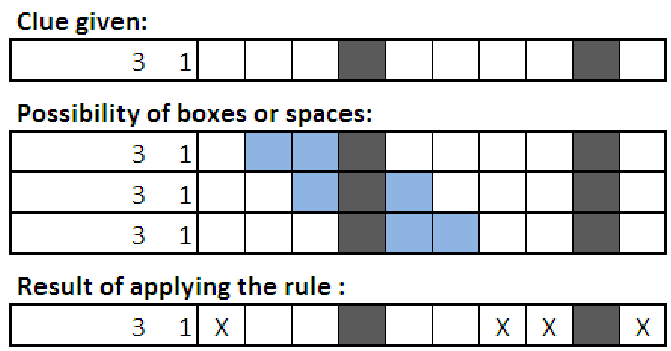


Figure 3. Simple Spaces

C. *Forcing*

In this method, the significance of the spaces will be shown. A space placed somewhere in the middle of an uncompleted row may force a large block to one side or the other. Also, a gap that is too small for any possible block may be filled with spaces. The forcing rule can be depicted in the following figure:

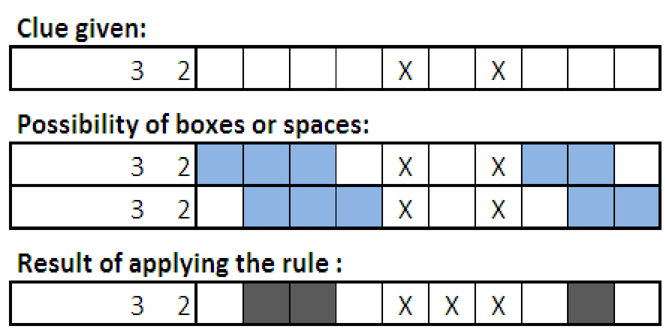


Figure 4. Forcing

D. *Contradiction*

Some more difficult puzzles may also require advanced reasoning. When all simple methods above are exhausted, searching for contradictions may help (Figure 5). It is wise to

ISSN 2301-6590

use a pencil (or other colour) for that in order to be able to undo the last changes. The procedure includes:

1. Trying an empty cell to be a box (or then a space).
2. Using all available methods to solve as much as possible.
3. If an error is found, the tried cell will not be the box for sure. It will be a space (or a box, if space was tried).

The problem of this method is that there is no quick way to tell which empty cell to try first. Usually only a few cells lead to any progress, and the other cells lead to dead ends. Most worthy cells to start with may be:

- cells that have many non-empty neighbours;
- cells that are close to the borders or close to the blocks of spaces;
- cells that are within rows that consist of more non-empty cells.

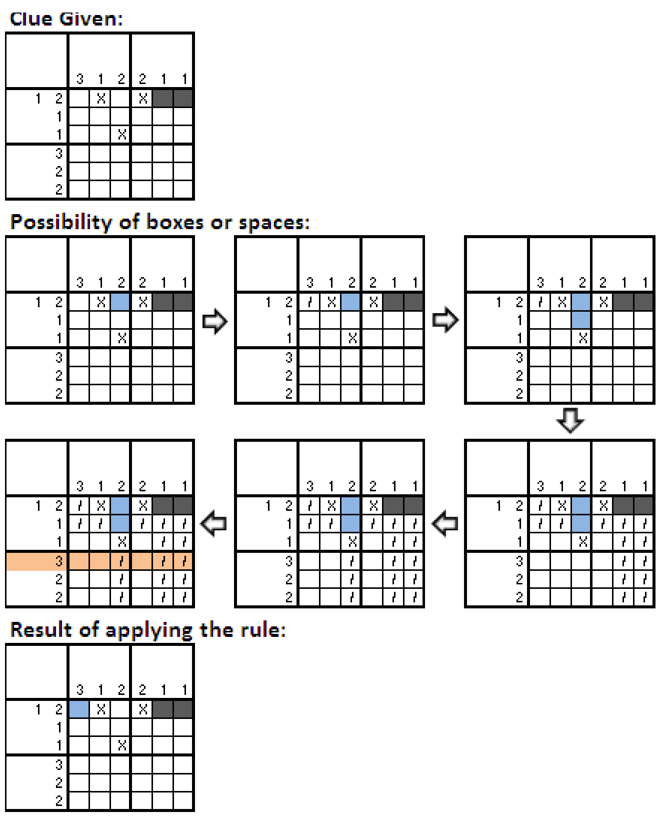


Figure 5. Contradiction

III. MODELLING OF PIXEL PUZZLE

The size of pixel puzzle is represented by b x k, where b isnumber of rows and k is number of columns respectively.Both are positive integer. In this paper, b and k are 10, 15, 20or 25. Each number clue in the left side of cells is denotedby $b(u,v)$, which means v_{th}component of u_{th}row. Eachnumberclue in the top is symbolized by $k(m,n)$, which meansm_{th}component of n_{th} column. According to the rule of pixelpuzzle, each component in rows and column must beseparated by empty spaces. Thus, there is a maximumcomponent for each rows and column. The maximumcomponent of v for u_{th}row, denoted by $v_{max}(u)$, equal to$\lceil k/2 \rceil$. In the same way, the maximum component of m forn_{th} column, denoted by$m_{max}(n)$, equal to $\lceil b/2 \rceil$. Accordingtothe following figure, the value of u, v, n and m, can bedefined more detail in the following way:

$u=1,2,\ldots,b$ $\quad$ $n=1,2,\ldots,k$
$v=1,2,\ldots,v_{max}$ $\quad$ $m=1,2,\ldots,m_{max}$

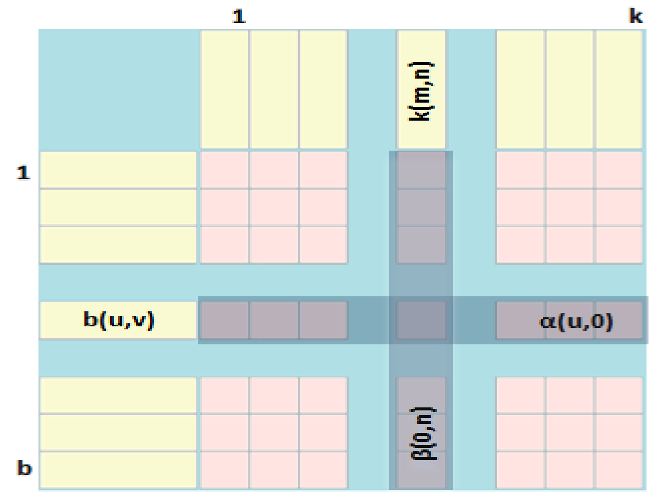


Figure 6. General representation of Pixel Puzzle

In the previous figure, $\alpha(u,0)$ and$\beta(0,n)$ are$(1\times k)$ and $(1\times b)$ row vectors respectively.

Each block in each row andcolumn is separated by one or more spaces. A number ofboxes for v_{th}component of u_{th}row is equal to *b(u,v)*. Due tothe fact that there are several possibilities of boxes, thepossible starting boxes for each *b(u,v)* is defined. In thispaper, the first starting possible boxes of *b(u,v)* is symbolizedby *firstrow(u,v)* and the last starting possible boxes of *b(u,v)*is denoted by *lastrow(u,v)*. In the same way, the first and laststarting possible boxes for columns are denoted by*firstcolumn(m,n)* and *lastcolumn(m,n)*. *firstrow(u,v)*,*lastrow(u,v)*, *firstcolumn(m,n)* and *lastcolumn(m,n)* variablescan be defined in the following formula :

$$firstrow(u,v) = \left\{ v + \sum_{i=1}^{v-1} b(u,i) \quad otherwise \right.$$

$$lastrow(u,v) = k - \left[b(u,v) - 1 + \sum_{j=v+1}^{v_{max}(u)} (b(u,j)+1) \right],$$

$$firstcolumn(m,n) = \begin{cases} 1 & m = 1 \\ m + \sum_{j=1}^{m-1} k(j,n) & otherwise \end{cases}$$

$$lastcolumn(m,n) = b - \left[k(m,n) - 1 + \sum_{j=m+1}^{m_{max}(n)} (k(j,n)+1) \right]$$

After defining the first and starting possible boxes for rowsand columns, the vectors are composed containing all possiblevalues for each component in rows and columns. An elementof the vector which has the value of 1 means box, while anelement with the value of 0 means space. Functions $_k R \to R_2$ α : and $_b R \to R_2$ β : for row and column aredefined in the following way:

$$\alpha(u,v) = (x_1 \quad x_2 \quad \dots \quad x_{k-1} \quad x_k)$$

$$x_i = \begin{cases} 1 & firstrow(u,v) \le i \le lastrow(u,v) + b(u,v) - 1 \\ 0 & otherwise \end{cases}$$

$$\beta(m,n) = (y_1 \quad y_2 \quad \dots \quad y_{b-1} \quad y_b)$$

$$y_i = \begin{cases} 1 & firstcolumn(m,n) \le i \le lastcolumn(m,n) + k(m,n) - 1 \\ 0 & otherwise \end{cases}$$

For each row and column, there are also $\alpha(u,0)$ and$\beta(0,n)$ functions which are the result of the *puzzle*. Eachelement in $\alpha(u,0)$ dan$\beta(0,n)$ has the value of 1 for box, 0for space,and 2 for undefined cell. It can be concluded thatfor each *pixel puzzle*, given *b(u,v)* and *k(m,n)* where $u = 1,2,\dots,b$ and $n = 1,2,\dots,k$, $\alpha(u,0)$ dan$\beta(0,n)$shall meetthe following conditions:

$$\forall u,v \quad \alpha(u,v) \;\to\; \sum_{i=1}^{k} x_i = b(u,v)$$

$$\forall m,n \quad \beta(m,n) \;\to\; \sum_{j=1}^{b} y_j = k(m,n)$$

And therefore the following conditions hold.

$$\forall u \quad \alpha(u,0) = \sum_{i=1}^{v_{max}} \alpha(u,i)$$

$$\forall n \quad \beta(0,n) = \sum_{j=1}^{m_{max}} \beta(j,n)$$

A. *Simple Boxes*

The *simple boxes* rule for v[th] component of u[th] row can be specified as follows:

Given: $b(u,v)$, $\alpha(u,v) = (x_1 \quad x_2 \quad \dots \quad x_k)$, first = first index of $\alpha(u,v)$ element with value of 1, and last = last index of $\alpha(u,v)$ element with value of 1 :

$$\forall u,v,\ ((first + b(u,v) - 1) \ge (last - b(u,v) + 1)) \to$$

$$\forall x_i \in \alpha(u,0), ((last - b(u,v) + 1) \le i \le (first + b(u,v) - 1)) \to x_i = 1$$

$$\forall i, y_u \in \beta(0,i), ((last - b(u,v) + 1) \le i \le (first + b(u,v) - 1)) \to y_u = 1$$

The *simple boxes* rule for m th component of n th row can be specified as follows:

Given: $k(m,n)$

and(,) ( ) 1 2 b β *m n* = *y y*… *y*, first = first

index of β (*m*,*n*)

element with value of 1, and last = last

index of β (*m*,*n*)

element with value of 1 :

∀*m*,*n*, ((*first*+*k*(*m*,*n*)–1) ≥ (*last*–*k*(*m*,*n*)+1))→

∀ ∈ (0,),((− (,) +1) ≤ ≤ (+ (,) −1)→ =1 *ii* *y* β *n last k m n i first k m n y*

∀ , ∈ (,0),((− (,) +1) ≤ ≤ (+ (,) −1)→ =1 *n n* *i x* α *i last k m n i first k m n x*

B. *Simple Spaces*

If the i[th] component at $\alpha(u,v)$ is symbolized by $z(u,v,i)$ and the j[th] at $\beta(m,n)$ symbolized by $w(m,n,j)$, then the following holds.

1. For row :

$$\forall u,i \quad \sum_{v=1}^{v_{max}} z(u,v,i) = 0 \to z(u,0,i) = 0 \wedge w(j,i,u) = 0,\ \ j = 0,1,2,\dots,m_{max}$$

$$\forall u,i,\ \exists v,\ z(u,v,i) = 1 \ \wedge z(u,0,i) = 1 \to$$

$$(z(u,v,j) = 0,\ \ j = 1,2,\dots,i - b(u,v)) \wedge (z(u,v,j) = 0,\ \ j = i + b(u,v),\dots,k)$$

2. For column:

$$\forall n,i \quad \sum_{m=1}^{m_{max}} w(m,n,i) = 0 \to w(0,n,i) = 0 \wedge z(i,j,n) = 0,\ \ j = 0,1,2,\dots,v_{max}$$

$$\forall n,i,\ \exists m,\ w(m,n,i) = 1 \ \wedge w(0,n,i) = 1 \to$$

$$(w(m,n,j) = 0,\ \ j = 1,2,\dots,i - k(m,n)) \wedge (w(m,n,j) = 0,\ \ j = i + k(m,n),\dots,b)$$

C. *Forcing*

If the i[th] component at $\alpha(u,v)$ is symbolized by $z(u,v,i)$ and the j[th] at $\beta(m,n)$ symbolized by $w(m,n,j)$, then the following holds.

1. For row :

$$\forall u,v,i \quad z(u,v,i)=0 \wedge i \le b(u,v) \rightarrow z(u,v,j)=0, \quad j=1,2,\ldots,i-1$$

$$\forall u,v, \ \exists i,j, \quad z(u,v,i)=0 \wedge z(u,v,j)=0 \wedge ((i-j) \le b(u,v)) \rightarrow$$
$$z(u,v,p)=0, \quad p=i+1,i+2,\ldots,j-2,j-1$$

$$\forall u,v,i \quad z(u,v,i)=0 \wedge ((i+b(u,v))>k) \rightarrow z(u,v,j)=0, \quad j=i+1,i+2,\ldots,k$$

If s = first index of $\alpha(u,v)$ element with value of 1 and e = last index of $\alpha(u,v)$ element with value of 1 :

$$\forall u,v, \ \sum_{i=1}^{k} z(u,v,i)=b(u,v) \rightarrow$$
$$z(u,p,q)=0, \ 1 \le p < v, \ q=s-1,s,\ldots,k$$
$$z(u,r,t)=0, \ v < r \le v_{\max}, \ t=1,\ldots,e+1$$

2. For column:

$$\forall m,n,i \quad w(m,n,i)=0 \wedge i \le k(m,n) \rightarrow w(m,n,j)=0, \quad j=1,2,\ldots,i-1$$

$$\forall m,n, \ \exists i,j, \quad w(m,n,i)=0 \wedge w(m,n,j)=0 \wedge ((i-j) \le k(m,n)) \rightarrow$$
$$w(m,n,p)=0, \quad p=i+1,i+2,\ldots,j-2,j-1$$

$$\forall m,n,i \quad w(m,n,i)=0 \wedge ((i+k(m,n))>b) \rightarrow w(m,n,j)=0, \quad j=i+1,i+2,\ldots,b$$

If s = first index of $\beta(m,n)$ element with value of 1 and e = last index of $\beta(m,n)$ element with value of 1 :

$$\forall m,n, \ \sum_{i=1}^{b} w(m,n,i)=k(m,n) \rightarrow$$
$$w(p,n,q)=0, \ 1 \le p < m, \ q=s-1,s,\ldots,b$$
$$w(r,n,t)=0, \ m < r \le m_{\max}, \ t=1,\ldots,e+1$$

D. *Contradiction*

This rule is applied when the simple boxes, simple spaces and forcing are not sufficient to solve the puzzle. Initially, *kemBaris*(u,v) as a set containing column index which possibly is a first starting possible boxes of $b(u,v)$ and *kemKolom*(m,n) as a set containing row index which possibly is a first starting possible boxes of $k(m,n)$ are defined. If $|kemBaris(u,v)|=1$ then $b(u,v)$ has only one possible solution. The same condition occurs if $|kemKolom(m,n)|=1$. Contradiction rule is applied when $|kemBaris(u,v)|>1$ or $|kemKolom(m,n)|>1$. Contradiction rule is implemented by trying an empty cell to be a box or a space. All possible rules are then applied to solve as much as possible. If an error is found, the tried cell will not be the box. However, it will be a space or a box.

E. *Best-First Search*

The best first search is performed if all logical rulesapplied have not given yet the solution. Basically, best-firstsearch is implemented initially by generating tree graduallyconsisting of components which have more than one solution.Then the logical rules are implemented repeatedly until allrows and columns component has one solution.

IV. EXPERIMENT RESULT

In this paper, there are three options to solve the puzzle,namely by using only Rule-Based method, only Best-FirstSearch method after applying Rule-Based method, and theexecution of Rule-Based continued by Best-First Searchmethod directly. Solving puzzle only by using Rule-Basedmethod means solving puzzle by employing *simple boxes*, *simple spaces*, *forcing*, and *contradiction* rules. The problems which can be and can not be solved by using only Rule-Based method are depicted in the following two figures respectively.

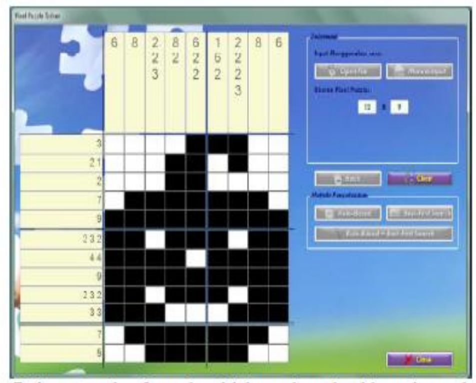


Figure 7. An example of puzzle which can be solved by using only Rule-Based method

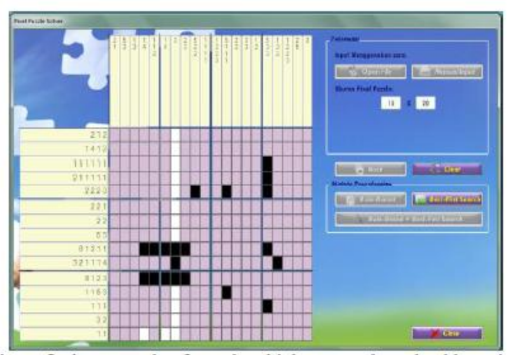
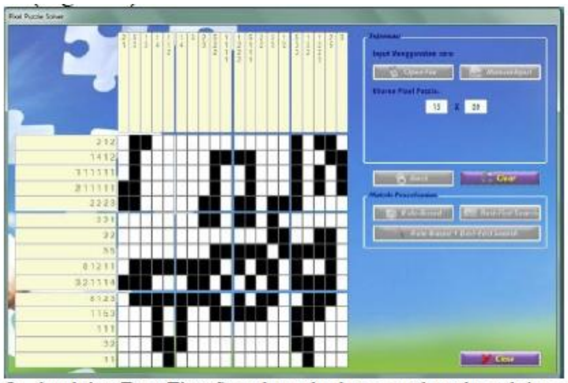


Figure 8. An example of puzzle which can not be solved by using only Rule-Based method

After the process of solving the above puzzle using Rule-Based method which can not be accomplished, solving the puzzle is then continued by implementing Best-first search method (Figure 9).

ISSN 2301-6590



igure 9. Applying Best-First Search method to complete the solving process

In the testing phase, 40 kinds of puzzle with 4 different sizes are analyzed. Table below gives the comparison of the time needed for solving puzzles by using only the Rule-Based method in milisecond.

Table IComparison Of The Time Needed For Solving Puzzles By Using Only The Rule-Based Method InMilisecond

Puzzle number	Puzzle size			
	10 x 10	15 x 15	20 x 20	25 x 25
1	16	25	30	951
2	20	31	54	1307
3	21	45	80	1805
4	23	60	327	1925
5	24	66	477	3829
6	26	78	1118	4553
7	28	89	1135	4568
8	29	271	1179	6710
9	31	303	2027	7626
10	63	397	3768	12654

According to the previous table, it can be seen that thedifferent problem with the same size can be solved in thedifferent time. The different problem with the different sizecan also be solved in the different time and the bigger sizedoes not always means the longer the time is needed to solve.Table above shows only the time needed to solve a puzzleby implementing only Rule-Based method. The time neededto solve a puzzle by using Rule-Based and Best-first search isnot analyzed because the usage of contradiction rule affectsvery few puzzles can be solved using *best-first search*. Mostof puzzles can be solved after implementing *contradiction* ruleand therefore in this paper the time needed to solve puzzlesusing Best-first search method is not analyzed further.In this section, it will be shown that in average, the longertime is needed to solve the bigger size puzzles using Rule-Based method. To conduct a testing about the average timeneeded to solve puzzles using Rule-Based method, *Student'sT-Test* is used with $\alpha = 5\%$. According to a series ofcomputation, it implies that individually, solving puzzle withbigger size is faster than puzzle with smaller size, but entirely,solving puzzle with smaller size is faster than puzzle with
bigger size.The second test is conducted to see whether solving onecell in puzzle with bigger size is longer according to thefollowing table:

TABLE 2
COMPARISON OF THE TIME NEEDED FOR EACH CELL IN PUZZLES TO BE SOLVED
BY USING ONLY THE RULE-BASED METHOD IN MILISECOND

Puzzle number-	Puzzle size			
	100	225	400	625
1	0.160	0.111	0.075	1.522
2	0.200	0.138	0.135	2.091
3	0.210	0.200	0.200	2.888
4	0.230	0.267	0.818	3.080
5	0.240	0.293	1.193	6.126
6	0.260	0.347	2.795	7.285
7	0.280	0.396	2.838	7.309
8	0.290	1.204	2.948	10.736
9	0.310	1.347	5.068	12.202
10	0.630	1.764	9.420	20.246

By implementing *Student's T-Test* with $\alpha = 5\%$, according to a series of computation, the average time to solve each cell is linear with the size of *pixel puzzle*, except for 10 × 10 and 15 × 15.

V. CONCLUSIONS AND FUTURE WORKS

In summary, we have shown the system which is able to solve pixel puzzle using Rule-Based Techniques and Best First Search. The Rule-Based Techniques applied consist of *simple boxes*, *simple spaces*, *forcing*, and *contradiction*. Two kinds of testing are performed by analyzing the time needed in milisecond for solving puzzles by using only the Rule-Based method. The first test shows that the bigger the puzzle, the longer time needed to solve the puzzle.

The second test shows that the average time needed to solve each cell in puzzle is linear with the size of *pixel puzzle*, except for 10 × 10 and 15 × 15. Problems which need to be solved with *best-first search* is done longer than problems which is solved with only Rule-Based technique because of heuristics process performed repeatedly. In the future, the system can be extended by categorizing difficulty level of problems and applying other method to enhance Rule-Based technique.

ACKNOWLEDGMENT

We would like to thank Coordinator of Research Development & Community Services in Computer Science

Faculty,UniversitasPelitaHarapan and Director of ResearchDevelopment & Community Services University of PelitaHarapan for the funding research support (funding code P003-FIK/II/2012).

REFERENCES

[1] (2011) ANONIM, NONOGRAM SOLVER [ONLINE] AVAILABLE: HTTP://WWW.GRIDDLER.CO.UK/SOLVE.ASPX

[2] (2011) ANONIM, THE GREAT NONOGRAM HUNT [ONLINE] AVAILABLE: HTTP://RAVENSPOINT.WORDPRESS.COM/2010/06/15/THE-GREAT-NONOGRAM-HUNT/

[3] (2011) ANONIM, WHAT IS NONOGRAM?. [ONLINE] AVAILABLE: HTTP://WWW.SUDOKU-PUZZLES.NET/NONOGRAM-PUZZLES/NONOGRAM/WHAT-IS-NONOGRAM.HTML

[4] MIN-QUAN JING, CHIUNG-HSUEH YU, HUI-LUNG LEE, LING-HWEI CHEN, ―SOLVING JAPANESE PUZZLES WITH LOGICAL RULES AND DEPTH FIRST SEARCH ALGORITHM,‖ *PROCEEDINGS OF THE EIGHTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNETICS, BAODING*, 2009.

[5] SALCEDO-SANZ, S., ORTIZ-GARCIA, E.G., PEREZ-BELLIDO, A.M., PORTILLA-FIGUERAS, A., XIN YAO, ―SOLVING JAPANESE PUZZLES WITH HEURISTICS‖, IEEE SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE AND GAMES (CIG 2007), 2007.