Fog of Search Resolver for Minimum Remaining Values Strategic Colouring of Graph

Saajid Abuluaih[1(✉)], Azlinah Mohamed[1,2(✉)], Muthukkaruppan Annamalai[1,2(✉)], and Hiroyuki Iida[3(✉)]

[1] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia
saajid.59@gmail.com, {azlinah,mk}@tmsk.uitm.edu.my
[2] Faculty of Computer and Mathematical Sciences, Advanced Analytic Engineering Center (AAEC), Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia
[3] School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Ishikawa 923-1292, Japan
iida@jaist.ac.jp

Abstract. Minimum Remaining Values (MRV) is a popular strategy used along with Backtracking algorithm to solve Constraint Satisfaction Problems such as the Graph Colouring Problem. A common issue with MRV is getting stuck on search plateaus when two or more variables have the same minimum remaining values. MRV breaks the tie by arbitrarily selecting one of them, which might turn out to be not the best choice to expand the search. The paper relates the cause of search plateaus in MRV to 'Fog of Search' (FoS), and consequently proposes improvements to MRV to resolve the situation. The improved MRV+ generates a secondary heuristics value called the Contribution Number, and employs it to resolve a FoS. The usefulness of the FoS resolver is illustrated on Sudoku puzzles, a good instance of Graph Colouring Problem. An extensive experiment involving ten thousand Sudoku puzzles classified under two difficulty categories (based on the Number of clues and the Distribution of the clues) and five difficulty levels (ranging from Extremely Easy to Evil puzzles) were conducted. The results show that the FoS resolver that implements MRV+ is able to limit the FoS situations to a minimal, and consequently drastically reduce the number of recursive calls and backtracking moves that are normally ensued in MRV.

Keywords: Fog of Search · Search plateau · Constraint satisfaction problem Graph colouring problem · Minimum remaining values · Contribution number Sudoku puzzles

B. W. Yap et al. (Eds.): SCDS 2018, CCIS 937, pp. 201–215, 2019.
https://doi.org/10.1007/978-981-13-3441-2_16

1 Introduction

Backtracking (BT) algorithms are widely adopted for solving Constraint Satisfaction Problems (CSPs), which includes the Graph Colouring Problem [1]. BT algorithm builds partial solutions (variable assignments) recursively in a process called 'labelling', and abandons an assignment as soon as it fails to be part of a valid solution in process called 'backtracking'. In order to improve the performance of the brute-force algorithms, heuristic strategies are applied to dynamically reorder the variables for labelling [2]. The idea is to reduce the number of backtracking to a minimum, and the Minimum Remaining Value (MRV) strategy is an efficient and popular strategy to that [4].

MRV deliberately selects a variable with the smallest domain size (least number of values) to expand a heuristic search. It uses Forward-Checking (FC) strategy to check in advance whether a labelling is doomed to fail.

A problem arises when MRV nominates more than one variable (with same minimum remaining values) for labelling because its heuristics is incapable of distinguishing the most promising variable for labelling. This uncertainty often arises in MRV due to what we call 'Fog of Search' (FoS) that is attributed to inadequacy of information to make a definite decision. Consequently, the paper presents a FoS resolver that helps to resolve FoS situations in MRV by employing an additional heuristics called the Contribution Number.

The rest of the paper is organised as follows. Section 2 describes the related works. The case study: Sudoku is presented in Sect. 3. The notion 'Fog of Search' is explained in Sect. 4. The improved MRV strategy called MRV+ that the FoS resolver implements is detailed in Sect. 5. The experiment and its results are discussed in Sect. 6, and finally Sect. 7 concludes the paper.

2 Related Works

The section briefly describe the key concepts that are related to our study, namely CSP, Graph Colouring and the MRV strategy, on which the proposed improvement is based.

2.1 Constraint Satisfaction Problem

Constraints Satisfaction Problem (CSP) is a well-studied example of NP-complete family of problems and exceedingly used by experts to model and solve complex classes of computational problems in artificial intelligence [3]. Finding a complete solution for this type of problems involves a search process to find a finite set of valid answers (or values) among the given candidates for a finite set of questions (or variables), without violating a finite set of restrictions (or constraints).

A CSP is defined mathematically by a triple (V, D, C) where V, D and C denote sets of variables, domains and candidates, respectively. $V = \{v_i\}_{i=1}^{n}$ is a finite set of n variables. Each variable v_i is associated with a set of potential candidates with which

the variable can be labelled with, i.e., the domain of v_i or $d(v_i)$. Consequently, $D = \{d(v_i)\}_{i=1}^{n}$ is a set of domains for each of the n variables. $C = \{c_i\}_{i=1}^{m}$ is a set of m constraints where each constraint $c_i = \langle s_i, r_i \rangle$ is a pair of relation r_i over a subset of variables $s_i = \{v_j\}_{j=1}^{k}$. The set of variables tightly associated through predefined constraints s_i, is normally referred to as 'peers'.

A CSP can be illustrated graphically as a Constraint Satisfaction Network such as one shown in Fig. 1, where $V = \{E, F, G, H\}$ whose respective domains are $d(E) = \{1, 2\}$, $d(F) = \{1, 3\}$, $d(G) = \{2, 4\}$ and $d(H) = \{2, 3, 4\}$. In this example, $\langle s', r' \rangle$ is an instance of a constraint c' involving variables $s' = \{F, H\}$ and relation $r' = \{F \geq H\}$. In this example, the constrained variables F and H are peers.

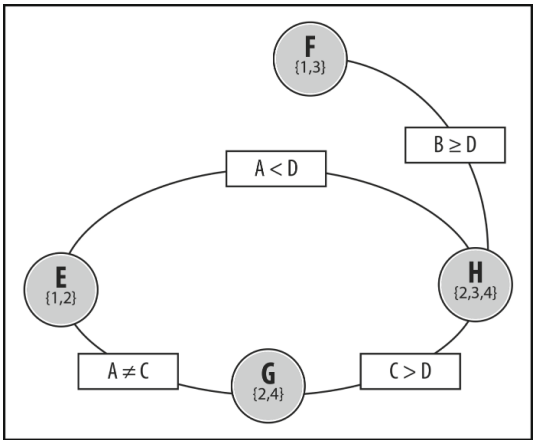

Fig. 1. An example constraint satisfaction network.

Graph Colouring Problem is a special subclass of CSP where the peers must not be labelled using same value; thus, only one type of logical relation, i.e., 'not equal to' ($\neq$) is applied to constrain the peers.

2.2 Solving Graph Colouring Problem

Solvers devoted to solving Graph Colouring Problems and CSPs in general can be classified under two main categories [4]: Deductive and Brute-force search algorithms.

Deductive search algorithm performs iterative improvements on variables' domains by implementing a set of deductive rules. The process of eliminating irrelevant values from a variable's domain emulates human deductive behaviour. At times, this approach fails to find a solution.

On the other hand, the brute-force BT search algorithm always finds solutions when there is one. In the worst case scenario it attempts all possible assignments on all unassigned variables until a solution is found or the possibilities ran out.

While the fundamental differences between these two approaches are significant, there are persistent efforts to merge their advantageous [5], which is also the aim of this paper.

A typical BT algorithm is incapable of ordering the variables right, which leads to trashing and affects the efficiency of the algorithm. As a consequence, the solving takes advantage of appropriate heuristics to prioritise the variables with the aim of pruning down the search space and to limit the occurrence of thrashing [6]. In place of a static variable order, a responsive selection mechanism that progressively reorders the variables as the problem solving process evolves, is often applied.

2.3 Minimum Remaining Values

The Minimum Remaining Values (MRV) strategy, which heuristically selects variable with fewest candidates to expand, is an existing popular strategy for Graph Colouring Problem [4]. MRV prioritises unassigned variables dynamically based on the number of available values that they hold, i.e., the candidates in the variables domains. According to its simple heuristics, the less the number of candidates in a variable's domain, the higher priority it receives as potential variable for search.

The counter-argument is if a variable with a large domain size is selected, the probability of assigning an incorrect candidate to it, is high. It could result in wasteful exploration of search space before the preceding bad assignment is realised. Poor variable selection also causes repeated failures.

On the contrary, MRV is a 'fail-first' heuristic strategy that helps to confirm if an assignment is doomed to fail at the early stages of the search. MRV applies Forward-Checking (FC) to preserve the valid candidates for the unassigned variables as the solving process progresses. While backtracking still occurs with MRV, it is considerably reduced by FC. As a result, the MRV strategy has been shown to accelerate the problem solving process by factor of more than a thousand (1000) times compared to static or random variable selection [4].

3 Sudoku as Case Study

Sudoku is a common Graph Colouring Problem example. It is a logic-based, combinatorial number-placement puzzle that has become popular pencil and paper game [14]. It is also regarded as a good example of difficult problems in computer science and computational search.

Basically, Sudoku is a group of cells that composes a board or also known as 'main grid' (see Fig. 2(a)). The main grid consists of a $\times$ b boxes or also known as sub-grids. Each sub-grid has a cells on width (row) and b cells on length (column). Subsequently, the main grid has a $\times$ b rows and a $\times$ b columns with a total of (a $\times$ b) $\times$ (a $\times$ b) cells (see Fig. 2(b)). The summation of the numbers placed in any row, column, or sub-grid is equal to the constant value of $\left(\sum_{i=1}^{a\times b} i\right)$, which in classical 9 $\times$ 9 board is equal to 45 [14]. In this paper, we consider the classical Sudoku board that has 9 (3 $\times$ 3) sub-grids, 9 rows, 9 columns, and 81 cells. The puzzle creator provides a partially

completed grid where some of these empty cells are pre-assigned with values known as 'clues' whereas the rest of the cells are left blank (see Fig. 2(c)). The objective is to place a single value out of a set of numbers {1, 2, 3…9} into the remaining blank cells such that each row, column and sub-grid contains all of the numbers 1 through 9 that total to 45 (see Fig. 2(d)).

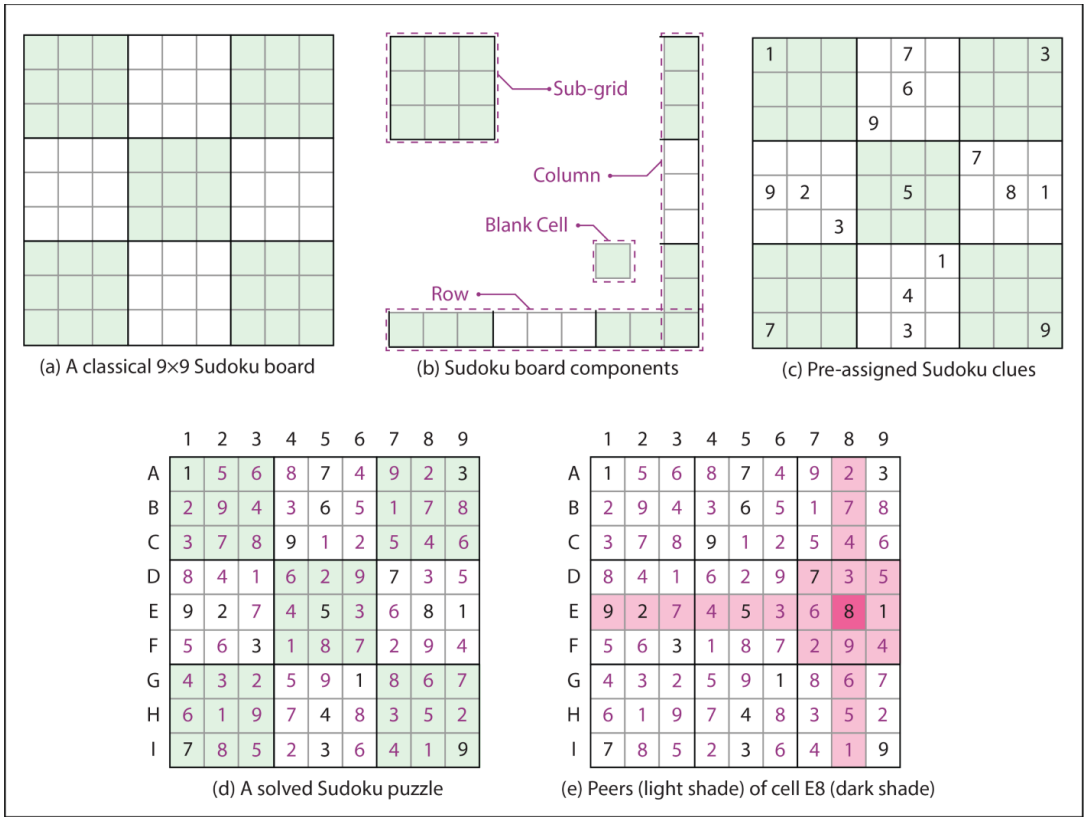

Fig. 2. Sudoku puzzle layout.

A 9×9 empty Sudoku board could generate $6,670 \times 10^{21}$ valid completed configurations. It has been shown that for creating a puzzles that has only one solution, at least 17 clues are required [15]. However, there is still no guarantee that a puzzle with 17 or more clues will have a unique solution [7, 8].

The constraints structure of classical Sudoku is such that each cell in the board is tightly associated with twenty (20) other cells or 'peers' (see Fig. 2(e)). Each cell has its own domain of potential values or candidates that can occupy the cell according to the candidates the peers are going to hold. The size of domain of a blank cell is 9, whose candidates are 1, 2, .., 9. Therefore, the domain of cell k, D_k can be defined mathematically as shown in Eq. 1, where DR_k, DC_k, and DS_k are sets of candidates of the assigned peers located on row, column, and sub-grid of cell k, respectively.

$$D_k = \{1, 2, .., 9\} \backslash \{DR_k \cup DC_k \cup DS_k\} \quad (1)$$

4 Fog of Search (FoS)

In military operations, gathering intelligence during on-going combats could be a serious challenge since the operational environment is partially observable and it is very hard to tell what is going on the other side beyond the visible distance [10]. The Prussian general Carl von Clausewitz realized that in all military conflicts, precision and certainty are unattainable goals and he introduced the term 'Fog of War' to describe this phenomenon [11]. During such situations, commanders rely on however small, incomplete and imperfect information that has been gathered to make 'intelligent' decisions, which is better than making spontaneous decision that leads to unpredictable outcomes.

Similarly, strategies devoted to improve CSP algorithms confront a sort of confusion when the best available choices have same heuristics value; a phenomenon known as 'search plateau' [12]. In the MRV strategy for instance, the search reaches a plateau if there are two or more variables holding the same minimum number of values or candidates, i.e., when the strategy is unable to discern the most variable among the them. We coined the term 'Fog of Search' (FoS) to express this state of confusion that hampers a strategy from progressing deterministically.

Consider the constraint satisfaction network shown in Fig. 1. Solving it using MRV will confront FoS at the very first assignment! Except for variable H that has three candidates in its domain, the rest of the variables have two candidates each. In this case, the strategy will exclude H from 'next variable to be solved' list, but a FoS situation arises because there are three other variables that hold the same minimum remaining value of 2. MRV is not designed to deal with the kind of ambiguity associated with FoS. In the example, MRV simply breaks the tie by selecting one of the variables E, F or G, in an arbitrary manner. When a strategy resorts to random actions to advance, it only tells the strategy is incompetent to deal with the problem, FoS in this case.

We know that the order in which the algorithm selects the 'next variable to be solved' significantly impacts its search performance. The question to ask is: can MRV make use the available information to break the tie among the best available choices in a FoS situation? The paper answers this question and proposes an improvement to MRV to help avoid the arbitrary selection of variable that occurs.

5 Minimum Remaining Values Plus (MRV+)

Typically, the MRV strategy iterates through all unassigned variables in a CSP, and compares each of their domain sizes before selecting a variable with the minimum remaining values or candidates as the 'next variable to be solved'. If there is only one variable with the optimal heuristic value, the variable is selected and labelled, and the search continues. However, if there is a FoS, MRV deals with the situation in two ways: (a) select a variable based on pre-assigned static order, i.e., the first variable

found with the desired optimal heuristic value will become the new frontier even if there are other variables holding the same heuristic value; (b) select a variable randomly among a group of variables that hold same optimal heuristic value.

Applied to solving Sudoku puzzles, the first approach is described by Algorithm 1 where the first cell that is found to have the minimum remaining value will be selected (see lines 6–9), while Algorithm 2 describes the second approach where one of the variables with minimum remaining value is selected randomly (see lines 23–25).

What is common with both approaches is the arbitrary selection of variables that does not effectively help to advance the search.

Algorithm 1 MRV Static Cell Selection

```
1:  Procedure SelectNextState()
2:    LessMRV ← int.MaximumValue
3:
4:  Fore ach Cell in SudokuBoard do
5:    If Cell.Value = NotAssignedCell AND
      Cell.Candidates.Count < LessMRV Then
6:      potentialCell ← Cell
7:      LessMRV ← Cell.Candidates.Count
8:    ENDIF
9:  End Fore ach
10:
11: If potentialCell = null Then
12:   //The current partial solution is inconsistent. Backtracking has to be committed.
13:   Return null
14: Else
15:   Return potentialCell
16: ENDIF
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
```

Algorithm 2 MRV Random Cell Selection

```
1:  Procedure SelectNextState()
2:    LessMRV ← int.MaximumValue
3:    PotentialCellsList ← null
4:
5:  Fore ach Cell in SudokuBoard do
6:    If Cell.Value = NotAssignedCell AND
      Cell.Candidates.Count < LessMRV Then
7:      PotentialCellsList ← NewList()
8:      PotentialCellsList.Add(Cell)
9:      LessMRV ← Cell.Candidates.Count
10:   Else If Cell.Value = NotAssignedCell AND
          Cell.Candidates.Count = LessMRV Then
11:       PotentialCellsList.Add(Cell)
12:     ENDIF
13:   ENDIF
14: End Fore ach
15: If PotentialCellsList.count < 1 Then
16:   //The current partial solution is inconsistent.Backtracking has to be committed.
17:   Return null
18: ENDIF
19: If PotentialCellsList.count = 1 Then
20:   //No FoS has been encountered, return the first cell in the list.
21:   Return potentialCellsList[0]
22: ENDIF
23: if PotentialCellsList.count > 1 Then
24:   //The strategy faces FoS. Return random cell .
25:   Return potentialCellsList[random(0, potentialCellsList.count)]
26: ENDIF
```

The paper proposes to involve a secondary heuristics strategy that is invoked upon detection of FoS in MRV. New heuristic values are generated to re-evaluate each of the indeterminate MRV choice variables. The secondary heuristics proposed is called Contribution Number (CtN), and it takes advantage of existing information to resolve FoS. Technically, the MRV+ strategy comprises MRV and CtN.

The CtN heuristics work by identifying the variables that have potentially valid candidates in common with their peers. The more candidates in common a variable has with respect to its peers, the greater is its contribution number. The argument is that by labelling a variable with the highest contribution number, will result in the deduction of most number of candidates from its peers' domains; thus, solving the problem quickly by hastening fail-first. Therefore, when MRV encounters a FoS, the 'next variable to be solved' is the one with minimum remaining value and maximum contribution number. The mathematical definition of the contribution number of variable k, CtN_k is given by Eq. 2 where U_k denotes the set of unassigned peers associated with variable k, and D_k is the domain of variable k. The function iterates through the domains of each unassigned peers of variable k and counts the number of candidates they have in common.

$$CtN_k = \sum_{i=1}^{|U_k|} \sum_{j=1}^{|D_i|} \left(d_j \in D_k\right) \tag{2}$$

$$U_k = P_k \backslash \{AR_k \cup AC_k \cup AS_k\} \tag{3}$$

In the context of Sudoku, the set of unassigned peers associated with variable of cell k is described by Eq. 3, where P_k is the set of peers associated with cell k (in the case of classical Sudoku P_k consists of 20 peers), and AR_k, AC_k, and AS_k are sets of assigned cells located on row, column, and sub-grid of cell k, respectively.

Algorithm 3 describes the application of MRV+ to Sudoku. When MRV detects FoS, the 'FogResolver' function that implements CtN is invoked (see line 25). The function receives a list of cells with same minimum remaining values (PotentialCellsList) as argument, and evaluates each cell's contribution number (see lines 31–41). Finally, the cell with the maximum contribution number is selected as a new frontier for MRV+ to explore.

Algorithm 3 MRV+

```
1:  Procedure SelectNextState()
2:    LessMRV ← int.MaximumValue
3:    PotentialCellsList ← null
4:
5:  Foreach Cell in SudokuBoard do
6:    If Cell.Value = NotAssignedCell AND
      Cell.Candidates.Count < LessMRV Then
7:      PotentialCellsList ← NewList()
8:      PotentialCellsList.Add(Cell)
9:      LessMRV ← Cell.Candidates.Count
10:   Else If Cell.Value = NotAssignedCell AND
          Cell.Candidates.Count = LessMRV Then
11:       PotentialCellsList.Add(Cell)
12:   ENDIF
13:   ENDIF
14: End Foreach
15: If PotentialCellsList.count < 1 Then
16:   //The current partial solution is inconsistent.Backtracking has to be committed.
17:   Return null
18: ENDIF
19: If PotentialCellsList.count = 1 Then
20:   //No FoS has been encountered, return the first cell in the list.
21:   Return potentialCellsList[0]
22: ENDIF
23: if PotentialCellsList.count > 1 Then
24:   //The strategy faces FoS. Return the most promising one.
25:   Return FogResolver(potentialCellsList)
26: ENDIF
27: Procedure FogResolver(PotentialCellsList)
28:   SelectedCell ← null
29:   CtNOfSelectedCell ← 0
30:
31: Foreach Cell in PotentialCellsList do
32:   Cell.CtN ← 0
33:   Foreach Peer in Cell.Peers do
34:     If Peer.Value = NotAssignedCell Then
35:       Foreach PeerCandidate in Peer.Candidates do
36:         If PeerCandidate is In Cell.Candidate Then
37:           Cell.CtN ← Cell.CtN +1
38:         ENDIF
39:       End Foreach
40:     ENDIF
41:   End Foreach
42:
43:   If Cell.CtN > CtNOfSelectedCell Then
44:     SelectedCell ← Cell
45:     CtNOfSelectedCell ← Cell.CtN
46:   ENDIF
47: End Foreach
48:
49: Return SelectedCell
50:
51:
52:
53:
54:
```

Figure 3(a) illustrates the dense distribution of large domain sizes for an instance of a difficult Sudoku puzzle. The darker a shaded cell, the larger is its domain size, i.e., it holds more candidates compared to a lightly shaded cell. In this illustrated example, MRV identifies three cells with same minimum remaining values of two candidates in each, namely D1{6,9}, E6{4,8}, and I2{8,9}. These cells are the most lightly shaded cells on the board. Among them, MRV+ selects D1 as the most promising choice to start the search because it has the maximum contribution number among them; $CtN_{D1} = 20$, $CtN_{E6} = 15$ and $CtN_{I2} = 14$. In this case, the candidates in the domain of D1 $\{6, 9\}$ also appear in twelve of its peers' domains: D2$\{6, 8, 9\}$, D3$\{6, 8, 9\}$, D5$\{4, 5, 6, 9\}$, D7$\{3, 5, 6, 8\}$, D8$\{3, 5, 6, 8\}$, C1$\{1, 6, 9\}$, F1, $\{1, 2, 6, 9\}$, G1$\{2, 4, 5, 6, 9\}$, H1$\{2, 4, 5, 6, 9\}$, I1$\{2, 5, 9\}$, E2$\{6, 7, 8\}$, and F3$\{1, 2, 6, 8, 9\}$.

Thus, labelling D1 before E6 or I2 will result in a greater reduction of the domain sizes in the problem. As the solving progresses, the domain sizes in the problem keeps shrinking until a solution is found, i.e., when the domain sizes of all unassigned cells is a singleton. Figure 3 graphically illustrates the dwindling unassigned cells and the concomitant shrinking of their domain sizes as the puzzle evolves from difficult to medium (Fig. 3(b)) to easy (Fig. 3(c)) problem in the solving process.

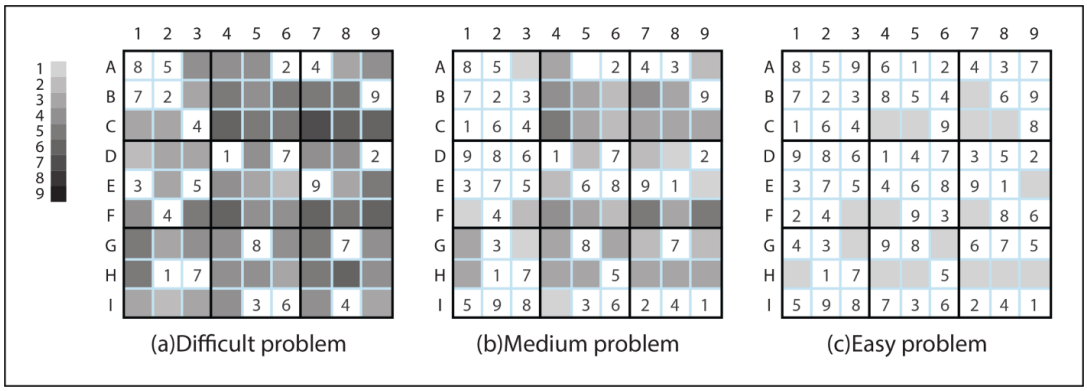

Fig. 3. The density and distribution of domain sizes of unassigned cells in a 9 × 9 Sudoku problem being solved. The darkest cell has nine candidates while the lightest cell has one candidate.

It is noteworthy to mention that the MRV+ strategy could still face FoS when the contribution numbers of the selected variables are same, in which case the tie at the second level has to be broken in an arbitrary manner. Such a situation happens when there is 'forbidden rectangle' or 'forbidden triangle' on a Sudoku board where the corner cells in the rectangle or triangle have same candidates [8], in which case their labels can be swapped. However, such incidents are rare. Moreover, these puzzles do not have unique solutions, and so are not regarded as valid puzzles.

6 Experiments, Results and Discussion

As part of our empirical experiment to evaluate the performance of MRV+ in relation to MRV, we encoded a solver that implements the BT algorithm extended with the MRV and MRV+ heuristic strategies. The MRV code was adapted from Peter Norvig's Python program [14]. MRV+ incorporates the CtN code within MRV.

The purpose of the experiment is: (a) To determine the number of times the MRV and MRV+ strategies confront FoS. For this, we implemented a monitoring function that records information related to the FoS occurrence; and, (b) To compare the performance of MRV and MRV+ in terms of the number of recursion and backtracking operations executed to solve a puzzle.

6.1 Performance Measures

Recursion and Backtracking are common performance measures of BT algorithms.

In our study, Recursion denotes the steps involved in the labelling of a cell whose domain size is two or more (Note: Assignment of 'naked single' is not counted):

- Select a cell with minimum remaining values (randomly selected in the case of MRV and strategically selected in the case of MRV+), then assign to it a candidate chosen from its domain; and,
- Perform FC and update the domains of its peers.

Backtracking denotes the steps carried out to undo an inconsistent labelling that occurs when certain constraint(s) is/are violated:

- Undo the FC updates to the domains of the peers; and,
- Free the candidate previously assigned to the cell. Upon backtracking, if there are other candidates in the domain of the freed cell then the solver performs Recursion with a new candidate chosen from its domain, otherwise it performs Backtracking once more to return to a previous state.

It is important to mention that the concept of neutralisation [13] has been adopted in this study, where a puzzle is considered solved once the main-grid is neutralised, i.e., when that the remaining unassigned cells are all are all 'naked single' cells. The assignment of 'naked single' candidates is trivial.

6.2 Simulation Data

The experiment uses simulation as a way to gain empirical insight towards problem solving. The advantage of the simulation approach is that it can be quick and easy to analyse the performance of the heuristics strategies. Large number of random simulations will find results close to the real behaviour of the heuristics. For this, we generated ten thousand (10000) Sudoku puzzles under two difficulty categories: Number of Clues [15] and Distribution of Clues [9].

- Number of Clues. Five thousand (5000) puzzles are randomly generated based on the number of clues. These puzzles are further divided into five difficulty levels, each with one thousand (1000) puzzles. They are Extremely Easy (50–61 clues), Easy (36–49 clues), Medium (32–35 clues), Difficult (28–31 clues) and Evil (22-27 clues) puzzles.
- Distribution of Clues. Five thousand (5000) puzzles are randomly generated based on the distribution of clues on the main grid. These puzzles are also divided into five difficulty levels, each with one thousand (1000) puzzles. They are Extremely Easy (each row, column, and sub-grid contains 5 clues), Easy (each row, column, and sub-grid contains 4 clues), Medium (each row, column, and sub-grid contains 3 clues), Difficult (each row, column, and sub-grid contains 2 clues), and Evil (each row, column, and sub-grid contains 1 clue).

The Evil puzzles in the Number of Clues category, which have at least 22 clues, are comparably easier to solve than the Difficult puzzles in the Distribution of Clues category, which have exactly 18 clues. The Evil level puzzles in the Distribution of Clues category, which have exactly 9 clues, are much harder to solve, not only because of the scanty clues but also because they are sparsely distributed. It should be noted that puzzles with 16 clues or less cannot have a unique solution [8], which includes the Evil puzzles in the Distribution of Clues category. In such cases where there can be more than one solution, the solver has been programmed to stop after finding the first solution.

6.3 Performance of MRV and MRV+

Tables 1 and 2 list the average number of FoS encountered by MRV and MRV+ for the Sudoku puzzles generated based on the Number of Clues and the Location of Clues categories, respectively. The 5,000 puzzles under each category are organised according to their difficulty levels where each level comprises 1,000 puzzles.

Table 1. Average FoS in solving 5,000 Sudoku puzzles generated based on the Number of Clues for the MRV and MRV+ strategies.

Strategy	Difficulty level (1,000 puzzles in each level)				
	Ext. easy Clues: 50–61	Easy Clues: 49–36	Medium Clues: 35–32	Difficult Clues: 31–28	Evil Clues: 27–22
MRV	17.9	33.4	44.2	53.7	68
MRV+	0	0	0	1	3

Table 2. Average FoS in solving 5,000 Sudoku puzzles generated based on the Location of Clues for the MRV and MRV+ strategies.

Strategy	Difficulty level (1,000 puzzles in each level)				
	Ext. Easy Clues (5)	Easy Clues (4)	Medium Clues (3)	Difficult Clues (2)	Evil Clues (1)
MRV	18.8	34.5	45	53.9	68.2
MRV+	0	0	0	2	5

The performances of MRV and MRV+ are measured in terms of the number of Recursion (R) and the number of Backtracking (B) as described in Sect. 6.1. The average number of recursion and backtracking executed for solving the puzzles according to their difficulty levels are shown in Tables 3 and 4. Table 3 lists the results for the Sudoku puzzles generated based on Number of Clues, while Table 4 lists the results for puzzles generated based on the Location of Clues. The latter are harder to

Table 3. Number of Recursion and Backtracking in solving 5,000 Sudoku puzzles generated based on the Number of Clues for the MRV and MRV+ strategies.

Strategies	Difficulty level (1,000 puzzles in each level)									
	Ext. easy Clues: 50–61		Easy Clues: 49–36		Medium Clues: 35–32		Difficult Clues: 31–28		Evil Clues: 27–22	
	R	B	R	B	R	B	R	B	R	B
MRV	19	0	35.7	0.8	53.5	9	80	31.5	195	139
MRV+	9.3	0	27.5	0.5	47.3	5.7	63.7	16.8	103.8	50.3

solve because of fewer clues and their distributedness. Therefore, the number of Recursion and Backtracking (in Table 4) are necessarily higher compared to their counterparts in Table 3.

Table 4. Number of Recursion and Backtracking in solving 5,000 Sudoku puzzles generated based on the Location of Clues for the MRV and MRV+ strategies.

Strategies	Difficulty level (1,000 puzzles in each level)									
	Ext. easy Clues (5)		Easy Clues (4)		Medium Clues (3)		Difficult Clues (2)		Evil Clues (1)	
	R	B	R	B	R	B	R	B	R	B
MRV	20	0	36	1	61	16	160.5	107	284	223
MRV+	10.6	0	28	0.5	54	11	95	44	121	61.5

In Extremely Easy and Easy puzzles, MRV targets unassigned 'naked single' cells first where FoS does not arise; selecting any of the cells with a single minimum remaining value will not give cause to backtracking. The FC strategy eliminates the assigned value from the domains of the peers. As a result, most of the remaining unassigned cells will eventually become 'naked single' cells too. Recall that a puzzle is considered solved (neutralised) when all the unassigned cells are 'naked single' cells, so the search is deliberately halted before all the cells in the main-grid are filled.

Even in few cases where FoS occurs in Easy puzzles, we observe that there are at most two minimum remaining values, and MRV almost always picks the right value to label during Recursion, so there is little or no backtracks. This explains the nearly equal numbers of Recursion and FoS in the Extremely Easy and Easy puzzles, under both Difficulty categories.

In more complex Medium, Difficult and Evil puzzles where the domain sizes of the variables with minimum remaining values are much larger (i.e., up to a maximum of nine in many instances of Evil puzzles), there are relatively lesser FoS situations. However, the chances of committing to wrong labelling using values from a large domain, is high. The more mistakes the solver makes, the more backtracks and retries it must perform. For this reason, the numbers of Recursion and Backtracking in the complex puzzles have increased dramatically.

The experiment demonstrates that MRV+ out-performs MRV to drastically reduce the FoS situations. In fact, MRV+ encountered no FoS for the Extremely Easy, Easy and Medium puzzles in both Difficulty categories. Even for the more complex Difficult and Evil puzzles, the FoS in MRV+ is negligible. For example, in the Number of Clues category, MRV+ on the average encountered only 1 and 3 FoS situations for the Difficult and Evil puzzles. These counts are extremely small compared to 53.9 and 68 average FoS situations that MRV encountered for the corresponding difficulty levels (see Table 1). The result is consistent in the Location of Clues category (see Table 2).

The much fewer FoS in MRV+ compared to MRV indicates that the second level CtN heuristics is able to differentiate the unassigned cells according to their influences on their peers and decisively select a more promising cell in Recursion. Subsequently,

the numbers of Recursion and Backtracking are consistently lower for MRV+ compared to MRV (see Tables 3 and 4). The efficiency of MRV+ is more distinct in Difficult and Evil puzzles where the number of Recursion and Backtracking of MRV+ are less by more than half of that of MRV.

The results tabulated in Tables 3 and 4 are graphically illustrated in Figs. 4 and 5 graphically, respectively. The graphs illuminate that MRV+ has consistently low numbers of Recursion and Backtracking compared to MRV. The difference between their numbers of Recursion (see Figs. 4(a) and 5(a)) and their numbers of Backtracking (see Figs. 4(b) and 5(b)) are significant in the Difficult and Evil puzzles.

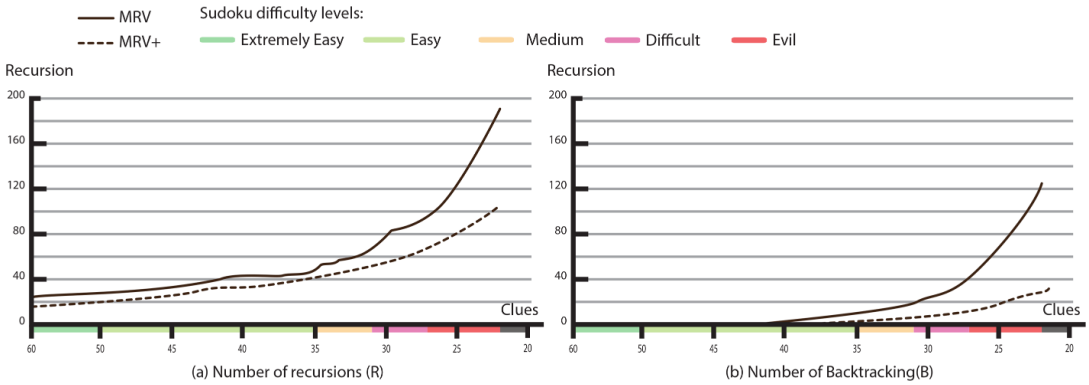

Fig. 4. Performance comparison between MRV and MRV+ for solving Sudoku puzzles generated based on the Number of Clues.

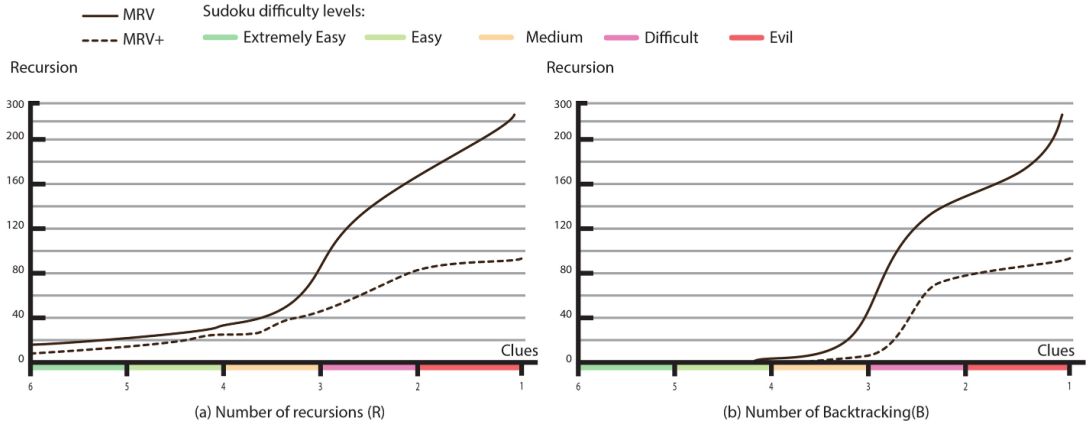

Fig. 5. Performance comparison between MRV and MRV+ for solving Sudoku puzzles generated based on the Location of Clues.

7 Conclusion

The Fog of Search (FoS) situation defines a state of confusion that a search strategy encounters when more than one variable shares the same optimal heuristic value. In the case of MRV, the optimal heuristics is the minimum remaining values, where the

common practice to select a variable arbitrarily. Moreover, the reason behind using heuristics in the first place is to rank the alternatives such that each gets rated based on how promising it is worth exploring given the limited resource (time, memory, and computing power) and being caught in a FoS means that the heuristic function has failed to achieve its purpose of design. Therefore, addressing FoS helps to overcome the failure of the existing heuristics.

The paper presents a secondary heuristics called Contribution Number (CtN) that enables MRV to make a resolute decision to resolve FoS. The function FogResolver implements the modified MRV+ strategy, which re-evaluates the choice variables that have same minimum remaining values (fewest number of candidates), then selects one that has greatest influence on its peers; the one with the maximum contribution number.

The results of an extensive experiment involving 10,000 puzzles under two difficulty categories and multiple difficulty levels show that MRV+ consistently outperforms MRV. The results indicate that the MRV+ strategy that fortifies MRV with CtN heuristics is resourceful in resolving FoS, and consequentially returns the solution with significantly lower number of Recursion and Backtracking than MRV.

In future work we plan to extend the application of CtN to value selection, i.e., to label the most promising variable with the most promising candidate as well, which we believe will further improve the efficiency of MRV+. We also plan to provide the proof of correctness of the generalised MRV+ for solving Graph Colouring Problems.

References

1. Poole, D.L., Mackworth, A.K.: Artificial Intelligence: Foundations of Computational Agents Artificial. Cambridge University Press (2010)
2. Edelkamp, S., Schrodl, S.: Heuristic Search: Theory and Applications. Morgan Kaufmann Publishers Inc. (2011)
3. Habbas, Z., Herrmann, F., Singer, D., Krajecki, M.: A methodological approach to implement CSP on FPGA. In: IEEE International Workshop on Rapid System Prototyping Shortening Path from Specification to Prototype (1999). https://doi.org/10.1109/iwrsp.1999.779033
4. Russell, S., Norvig, P.: Artificial Intelligence A: Modern Approach, 3rd edn. Pearson (2010)
5. Sudo, Y., Kurihara, M., Yanagida, T.: Keeping the stability of solutions to dynamic fuzzy CSPs. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 1002–1007 (2008)
6. Haralick, R.M., Shapiro, L.G.: The consistent labeling problem: Part I. IEEE Trans. Pattern Anal. Mach. Intell. 173–184 (1979). https://doi.org/10.1109/tpami.1979.4766903
7. Jilg, J., Carter, J.: Sudoku evolution. In: 2009 International IEEE Consumer Electronics Society's Games Innovations Conference, pp. 173–185 (2009). https://doi.org/10.1109/icegic.2009.5293614
8. Mcguire, G., Tugemann, B., Civario, G.: There is no 16-clue sudoku: solving the sudoku minimum number of clues problem via hitting set enumeration. Exp. Math. **23**, 190–217 (2014)
9. Jiang, B., Xue, Y., Li, Y., Yan, G.: Sudoku puzzles generating: from easy to evil. Chin. J. Math. Pract. Theory **39**, 1–7 (2009)
10. Kiesling, E.C.: On war without the fog. Mil. Rev. 85–87 (2001)

11. Shapiro, M.J.: The fog of war. Secur. Dialogue **36**, 233–246 (2005). https://doi.org/10.1177/0967010605054651
12. Asai, M., Fukunaga, A.: Exploration among and within plateaus in greedy best-first search. In: International Conference on Automated Planning Schedule, pp. 11–19 (2017)
13. Abuluaih, S., Mohamed, A.H., Annamalai, M., Iida, H.: Reordering variables using contribution number strategy to neutralize sudoku sets. In: International Conference on Agents Artificial Intelligence, pp. 325–333 (2015). https://doi.org/10.5220/0005188803250333
14. Norvig, P.: Solving Every Sudoku Puzzle (2010). http://www.norvig.com/sudoku.html
15. Lee, W.: Programming Sudoku, 1st edn. Apress (2006)