Improving the efficiency of forward checking algorithm for solving constraint satisfaction problems

Yusef Farhang
Department of computer science, Islamic Azad University
Khoy Branch, Iran
yfarhang@yahoo.com

M.R. Meybodi
Department of computer Engineering, Amirkabir University of Technology,
Tehran, Iran
mmeybodi@aut.ac.ir

A.R. Hatamlou
Department of computer science, Islamic Azad University
Khoy Branch, Iran
reza_hatamloo@yahoo.com

Abstract

There are various problems concerning artificial intelligence which can be stated in the from of constraint satisfaction problems (CSP). These problems are defined by a set of variables and a set of constraints in the range of selectable values for variables. Solution of this problem is a set of values for variables so that all constraints of the problem are satisfied. A part of Algorithms for CSP are forward algorithms. They are used to check consistency and constraint propagation. The most famous forward algorithm, is forward checking algorithm (FCA). In this article we are going to introduce FCA and suggest two algorithms to improve efficiency of forward.

1. Introduction

CSP covers a great number of problems about artificial intelligence(AI), among which we can name character puzzle, eight queens , graph coloring , image processing and circuit planning etc. The main indicators of CSP are variables and constraints. The basic goal in solving these problems is specifying the value for variables so that the whole constraints of problem are satisfied. We select the values from a specified domain. Solving the AI problem is accompanied by searching. CSP Algorithms include two great groups of consistency and search (back tracking) and local search algorithms. Again the back tracking algorithm is split into back ward and forward algorithms.

Consistency algorithms use constraints propagation. Arc consistency and maintaining are among consistency algorithms. In local search algorithms, the path of reaching the target is not important and there is no systematic problem space. It is a kind of partial algorithm i.e. it may achieve no answer. Hill climbing algorithm is one of local algorithms which fails in 86 percent of case and just solve the problem in 14 percent of case, however very fast. This algorithm gains an answer for the problem but just one answer not the entire. Local search algorithms act instead of some state path, and one transferred just to its neighbor state.

In back ward algorithms, constraint propagation is not used, but it is tried to move forward search action by estimates, consistent with previous constraints. In backward algorithms no consistency checking with future variables is checked to specify values for current variables. But all checking are performed with pervious variables. Backward search algorithms include standard back tracking, back checking and back marking.

In forward algorithms with constraints propagation, the domain becomes small. In that case, the number of processes and consistency checking becomes less. Forward algorithms achieve the answer faster than back ward algorithms. In forward algorithms, constraint propagation is more common and current and future variables are checked. We can introduce two examples in this category : Fully look ahead and partially look ahead Algorithms.

We suggest forward checking in order to improve efficiency of algorithm. At first, we discuss forward checking algorithms. Finally we will consider the result of these algorithms from view point of the number of consistency checking on the problem of n-queens.

2. The improved algorithm

This algorithm is an improved form of forward checking algorithm. They both generally act in a same

978-0-7695-3382-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ISDA.2008.272

way and start from null or empty. This algorithm minimizes the domain by constraint propagation and deleting the inconsistent values from future domain. This process continues until a variable empties and a back track occurs.

In this algorithm, the values are observed and if there is an unused one which cannot be used for current variable and future, the algorithm stops and back track happens (if a value is not used in a process fails and algorithm recognizes the situation and stops).Then if a value is inconsistent with all variables or is not chosen for any variable, the process stops. This algorithm lessens the processes, the number of back tracks and inconsistencies and solving time. It is used in symmetric problems, and also in problems that require a value for each variable i.e. the number of values and variable are the same. The following fig 1 is a pseudo code of the algorithm.

```
Procedure forward_checking_O1
While d(i) is not empty
Begin while
        Empty_domain ← false
        For each variable y in Xk (I<k<n)
        Begin for
                For value v in d(k)     (begin I+1)
                   Begin for
                        If not value <v,y> then
                        Begin if
                                Reset each d(k), I<k<=n to its state
before a was select.
                                Break;
                        End if
                 End for
                For value v in d(k)
                Begin for
                        If not consistent <y,v> then
                        Begin if
                                Call function fun
                                 Variable ← value
                                 Empty_domain =true
                                  Break;
                        End if
                 End for
          End for
          If empty_domain= false then
          Begin if
                 Reset each d(k), I<k<=n to its state before a was
select.
          End if

End while
Function fun
     Begin function
                For variable Xk (I+1<k<=n)
                   For each value v in d(k)
                              Give consistent
     End function
```

Fig.1. pseudo code of the suggested algorithm.

This algorithm finds all present answers and we use the first answer for n- queens. In this regard horizontal and vertical checking are performed.

3. solution of 8- queens problem by the first suggested Algorithm

In this section, we will describe the first suggested algorithm to solve the 8- queens problem. At first, it specifies value of first variable and then deletes all inconsistent values as forward checking algorithm. After that, all variables are checked. They will be given values if their domains are not empty, otherwise they will stop, back track and a new value will be selected for previous variable. The algorithm checks the values as well. If a value has not been used before and it is inconsistent for future variables, the process would have been stopped, and back tracked until a new value is chosen for previous variable.

In 8 queens problem, The first queen sits in first row and first cell and then all inconsistent cells of next rows are deleted. Afterwards, all rows are checked. Each row must have at least one empty cell (horizontal checking) and also unvalued columns are checked. They must have empty cells from current variable to the next one (vertical checking). Considering two mentioned checking, the queen is put into the first empty cell which is in second row and third cell. Then all inconsistent cells of other rows are omitted. The third and fourth queens are given values as stated above.

When the fourth queen is positioned, all future rows are checked. They must have at least one empty cell, however the sixth row has not. Therefore, solution stops and a back track happens while there is an empty cell in fifth row (the answer of horizontal checking was wrong and so back track happens). You can see the processes in following fig 2.

Q							
x	x	Q					
x	x	x	x	Q			
x	Q	x	x	x	x		
x	x	x		x	x	x	
x	x	x	x	x	x	x	x
x	x	x		x		x	x
x	x	x		x	x		x

Fig.2.

In next stage, a new value is selected for current variable which is the fourth row with the fourth queen and BT is occurs in previous value that is a back track. The fourth queen is in first empty cell in fourth row which is the seventh. Cells are deleted. Then the next

row appears and two vertical and horizontal checking are performed. Firstly, all the rows and presence at least an empty cell in any row is checked which is a right considered and tried to use them in future variables. When the unused value is inconsistent for all future variables, the algorithm stops. As you can see, the value of latter is inconsistent for all future variables. Therefore continuation of this algorithm is useless and leads to no answer. Backtrack is necessary, but in forward algorithm the process continues since there is an empty cell in next rows (in this section two processes of forward checking algorithm is reduced). The process is presented in fig 3.

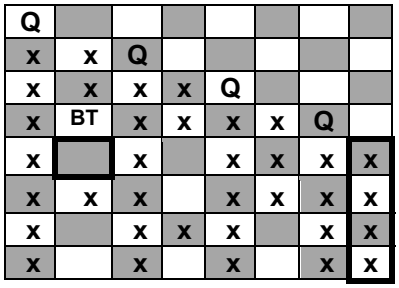

Fig.3.

In Third process once more a value is specified for current variable i.e. forth queen and BT substitute for previous value (the fourth queen is put in 8th cell). Then next row appears and inconsistent cells with fourth queen are deleted in next rows. Since the checking resulted in positive, the 5th queen sits in the first empty cell which is the second cell, and then inconsistent cells in next rows are deleted. After that the 6th and 7th queens are valued as previous one, horizontal checking is performed. Observing the lack of empty cell in 8th row a back track process goes on. (fig.4)

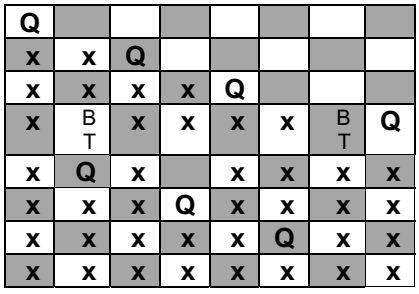

Fig.4.

This process continues in this way to solve the 8-queens problem by the first suggested algorithm. This algorithm can find all 98 answers of eight queens problem.

4. The second suggested Algorithm

This algorithm improves the efficiency of forward checking Algorithm too. It also compares just current and future variables, and propagate constraints. Second suggested algorithm, specifies the value for the first variable and then in next variables, inconsistent values are deleted until the first consistent value appears (despite of FCA, all inconsistent values are not deleted). Then all future variables are checked from view point of no empty of the domain otherwise, the algorithm stops, and backtrack appears: In horizontal checking, when the range is not empty, the first value is contact with previous variables. In this situation the value is devoted to current variable. Therefore the second suggested algorithm provides consistency checking as much as requires and prevents extra checking. We must note that in value giving to variables, the value to be used must not have inconsistency for previous variables. The algorithm finds all present answers of the problem.

It begins from empty situation and after specifying or giving a value to the first variable, goes toward next variable and constraints until the first consistency value in each variable. For instance in n-queens problem, the first queen is put in first variable and then goes to the next row and constraints are propagated. In the second variable, constraints are propagated until an empty cell finds. If there is an empty cell in a row, we go to next variable. This process continues until all variables are values. If an empty domain variable appears, we go toward the next variable. The same process continues until all variables own a value. Back track is written and another value is selected. The new value must be checked with previous variables in order to prevent inconsistency, when it proves its consistency, it is accepted and solution go on the difference between this algorithm and FCA is that in the latter all variables have constraints, but in the first one, constraints go on until a consistent value appears pseudo code of the second suggested algorithm is as follow fig 5.

```
Procedure forward_checking_O2
While d(i) is not empty
Begin while
        Empty_domain ← false
        For each variable y in Xk (I<k<n)
        Begin for
                For value v in d(k)
                Begin for
                        If not consistent <y,v> then
                        Begin if
                                Call function fun
                                 Variable ← value
```

```
                                    Empty_domain =true
                                     Break;
                            End if
                    End for
            End for
            If empty_domain= false then
            Begin if
                    Reset each d(k), I<k<n to its state before a was
select.
            End if

End while
Function fun
      Begin function
                    For variable Xk (I+1<k<=n)
                    Begin for
                          For each value v in d(k)
                          Begin for
                                If <v,y> empty
                                Begin if
                                   break
                                end if
                                Give consistent
                           End for
                    End for
      End function
```

Fig.5. pseudo code of the second suggested algorithm.

The second suggested algorithm lessens the required consistency checking considerably and even to one fourth of previous. It achieves all present answers of the problem. Of course in this article the first answer is used to compare.

5. solving the 4-queens problem using the second suggested Algorithm

In this algorithm, solving the 4-queens problem the first variable gains a value and then inconsistent values are deleted until the first consistent value appears. As soon as the first consistent value is found, searching for inconsistent value stops. In 4- queens problem the first queen is in first cell at beginning. (stage 1 in fig 6) After that inconsistent cells are deleted until the first empty cell appears. In second row, the third cell and in third row, the second cell and in fourth row, the second cells are empty. In the next process, the second queen gains value. From now we should pace attention to check empty cells have no contact with previous queens. After checking, the second queen gains a value. In next rows the cells containing queens contacts are deleted until the first consistency appears e.g. in third row, the second cell with second queen, the third cell with the first queen and the fourth cell with the second queen have contacts and must be deleted (stage 2 in fig 6).Since there is no empty row in this cell. Back track is performed (in second suggested algorithm, horizontal checking is performed for all future rows as in previous algorithm).

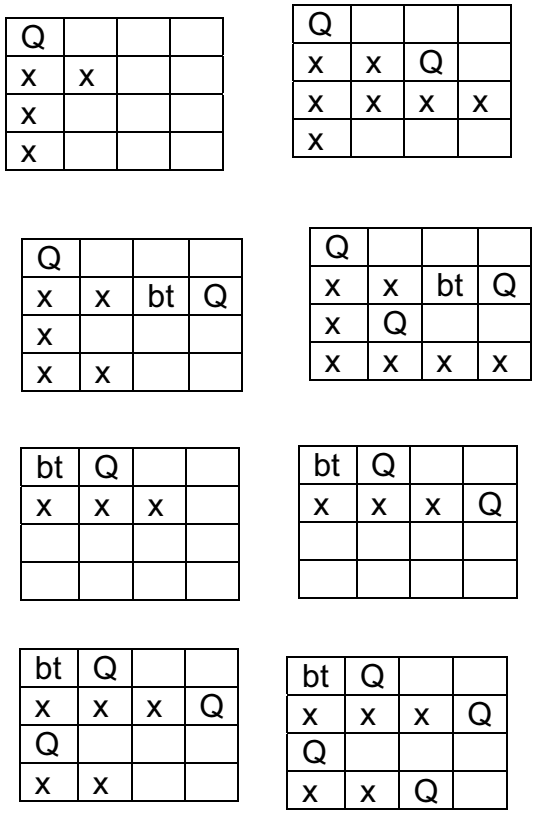

Fig.6. solving the problem of four queues by using the second suggested Algorithm.

In third stage another value is selected for the second variable. At the beginning, fourth cell is checked with previous queens in order to deny any contact.
Since there is no contact, the second queen occupies it, and inconsistent cells are deleted until appearance of first consistent cell. In third row, the first cell is consistency of second cell and checking stops. In fourth stage, the third row appears and the third queen gains a value. Since there is no empty cell after deletion of inconsistent cells in fourth row, the solution fails and doesn't continue and a back track appears.
In fifth stage the first row comes out and the value of the variable substitutes the previous value of BT because there is no any new value. Then we go to next rows and delete inconsistent cells until the first empty consistent cell appears. Then the second, third and fourth queens gain values as the first one (stages 6, 7 and 8 in fig 6).
In previous table, the 4-queens problem was solved, In this problem, the number of consistency checking is 15 while in FCA it was 38.

6. Implementation results

In this section, we consider the result of implementing the first and second suggested algorithms and also forward checking algorithm. We will discuss three categories of solution time, processes and the number of consistency checking. The time required to solve the problem of n- queens has been provided in fig 7 for all algorithms. The time of first suggested algorithm is less than two others. Therefore, it achieves the answer more rapidly. The time it takes to solve the problem is half the time that FCA takes to solve the same problem. The same is true about the second suggested algorithm and FCA. In fig 7 you can see the time required to solve the problem by each algorithm. Time decrease in first suggested algorithm is considerable and it achieves the answer very rapidly.

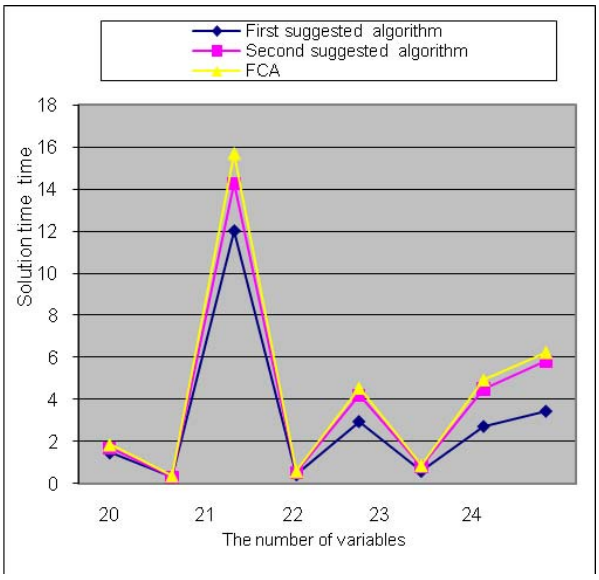

Fig.7.reviews the comparison graph by which the time of problem solution in three algorithms.

In fig 8 you can observe the number of processes to solve n-queens problem for all algorithms. As you see, the number in first suggested algorithm is half of the others; however, this number is the same in two other algorithms. The fig 8 shows a comparison in this regard.

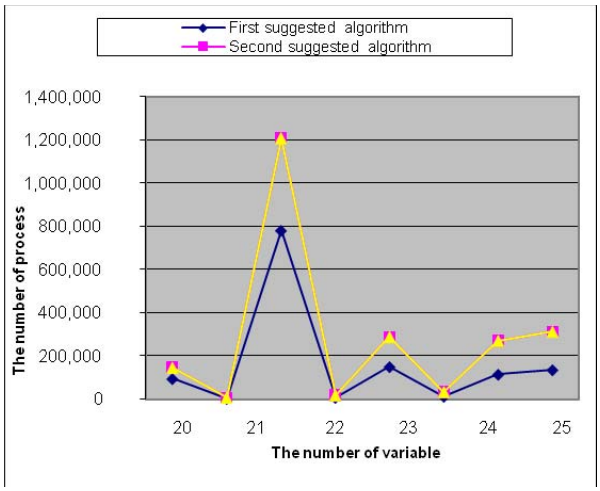

Fig.8.also shows the number of steps in which n queen problem is solved for three algorithms.

The numbers of consistency checking in solving the n-queens problem are provided in fig. 9. This number in second suggested algorithm is considerably less than two others, because unnecessary checking are deleted. A comparative graph of this criteria is provided in fig. 9.

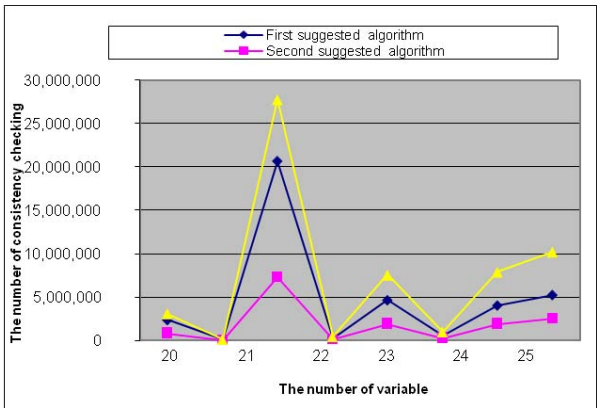

Fig.9. the comparison of number of consistency checking items in three FC, first suggested and second suggested algorithms

Comparing the number of consistency checking in three algorithms, we observe that the second suggested algorithm decreases this number from 1,236,589,062 to 289,307,109 which is very interesting. This advantage becomes more obvious if the solution of n- queens continues.

7. Conclusion

Two new algorithms are suggested to improve the efficiency of forward checking algorithm. They can find all answers of the problem and satisfy all constraints as FCA. The aim of these two algorithms is to decrease the number of consistency checking and processes in constraints satisfaction problem. The first suggested algorithm halves the number of consistency checking. Also, the second suggested algorithm decreases the number of consistency checking relative to FCA to one – fourth; therefore, the new algorithms have the advantage of decreasing the number of consistency checking. In this article, the suggested algorithms were studied from view point of n-queens problem solving and we suggest that you consider them in other problems. You can also check a combination of two suggested algorithms in future articles.

8. References

[1] VIPIN KUMAR, Algorithms for Constraint satisfaction problems: A surrey. *AI Magazine,* 32-44,1992.

[2] T.J.WARWICK, A GA Approach to constraint satisfaction problems, *Department of computer science university of ESSEX* Wivenhose park Colchester ESSEX, 1995.

[3] DANIEL HUNTER FORST, Algorithms and Heuristics for constraint satisfaction problems, *University of California Irvine*, 1997.

[4] FAHIEM BACCHUS AND ADAM GROVE, Looking forward in constraint satisfaction algorithms, January 22,1999.

[5] Greenwald, Constraint Satisfaction, *Artificial Intelligence*, Spring 2004.

[6] Luke Biewald, Tim Heilman, Mary Ann Brennan, Binary CSP Algorithms. May 13,2003.

[7] Fahiem Bacchus, Extending Forward Checking, *Department Of Computer Science*, University Of Toronto , 2001.

[8] David Mcallester, Constraint Satisfaction Search, *Artificial Intelligence* 1992.

[9] Michael J.Dent and Robert E.Mercer, An Empirical Investigation Of The Forward Checking Algorithm And Its Derivatives, *Proceedings of the 8th international conference on tools with Artificial Intelligence* (ICTAI 96), 1996 IEEE.

[10] O. Angelsmark and P. Jonsson, Improved algorithms for counting solutions in constraint satisfaction problem, *In CP 2003*, pages 81–95, 2003.

[11]. M. Caramia and P. Dell'Olmo, Constraint propagation in graph coloring, *Journal of Heuristics,* 8:83–107, 2002.