Comments on Mohr and Henderson's Path Consistency

Ching-Chih Han

Chia-Hoang Lee

Report Number:

88-733

COMMENTS ON MOHR AND HENDERSON'S PATH CONSISTENCY ALGORITHM

Ching-Chih Han
Chia-Hoang Lee

CSD-TR-733
January 1988

Comments on Mohr and Henderson's Path Consistency Algorithm

Ching-Chih Han
and
Chia-Hoang Lee

Department of Computer Science
Purdue University
West Lafayette, IN 47907

ABSTRACT

Mohr and Henderson have presented new algorithms for arc and path consistency in [1]. Though the underlying ideas of their algorithms are correct, the path consistency algorithm PC-3 is in error. In this paper we point out the errors in this algorithm and give a correct one. The time complexity and space complexity of the revised algorithm are also analyzed.

1. Introduction

In a recently published paper by Mohr and Henderson (M-H), new algorithms for arc and path consistency were presented. We show that the algorithm for path consistency is in error and give a refined solution.

The idea of the arc consistency algorithm introduced by Mackworth [2] is based on the notion of support. Mohr and Henderson further made this support evident by using some data structures to record the relevant supporting information. They use a counter for each arc-label pair $[(i,j),b]$ to indicate the number of labels at node j that support (are consistent with) the label b at node i. In addition, for each label c at node j, the member (i,b) of set S_{jc} is the label b at node i that is supported by label c at node j. They also use a table, M, to keep track of which labels have been deleted from which nodes, and a list, List, to control the propagation of constraints. This idea has also been applied to the path consistency algorithm PC-3 straightforwardly. However, the path consistency algorithm is not completely correct. In section 2, we will point out the errors and give a counterexample. In section 3, we will give a correct path consistency algorithm PC-4. Note that a formal treatment of the concept of path consistency was first provided by Montanari [4].

This work is supported by the National Science Foundation under the grant IRI-8702053.

2. A Counterexample to Algorithm PC-3

Arc consistency algorithms check the binary relation between each pair of nodes and delete any inconsistent labels from the admissible labeling set of each node. However, in path consistency algorithms, we cannot delete labels from the admissible labeling set of a node since we are considering the relations between two nodes instead of the labelings at a single node. To show that the algorithm PC-3 is not completely correct, let us examine the following example depicted in Figure 1. (Counter[*(i,j),b,c*] should be Counter[*(i,j),k,b,c*] on lines 26 and 27 of algorithm PC-3 in [1]) We will follow [2] and use matrix forms to represent binary relations.

$$N = \{1,2,3\}$$

$$A = \{b,c\}$$

$$E = \{(1,2),(1,3),(2,3)\}$$

$$A_1 = \{b,c\}$$

$$A_2 = \{b,c\}$$

$$A_3 = \{b,c\}$$

$$R_{12} = \begin{array}{cc} & \begin{array}{cc} b & c \end{array} \\ \begin{array}{c} b \\ c \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \end{array}$$

$$R_{13} = \begin{array}{cc} & \begin{array}{cc} b & c \end{array} \\ \begin{array}{c} b \\ c \end{array} & \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \end{array}$$

$$R_{23} = \begin{array}{cc} & \begin{array}{cc} b & c \end{array} \\ \begin{array}{c} b \\ c \end{array} & \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \end{array}$$

$$R_{ij} = R_{ji}^T$$

Note that matrix R_{ij} is the relation between i and j whose rows correspond to the possible labels for node i and columns to the possible labels for node j. (In algorithm PC-3 $R_{ij}(b,c)$ is represented by $R(i,b,j,c)$.) For the entries of R_{ij}, we use 1 to represent *true* and 0 to represent *false*. It is easy to see that (*c*, *b*, *b*) is the only solution of the above constraint satisfaction problem. However, after the first iteration of the **for** loop from line 2 to line 19 (see [1]), the data structures will be as follows:

$M[1,b] = 1$,
$M[2,b] = 1$,
All other M's are zero,
$A_1 = \{c\}$,
$A_2 = \{c\}$,
$A_3 = \{b, c\}$,
All S's are empty,
List $= \{(1, b), (2, b)\}$.

The two b's are removed from the sets A_1 and A_2 because there are no labels at node *3* that can support the relation *R(1, b, 2, b)*. But we know that *(c, b, b)* is the solution of the constraint satisfaction problem. If set A_2 does not contain label *b*, then *(c, b, b)* cannot be a final solution of the constraint satisfaction problem. The error comes from the deletion of label *b* from set A_2 at line 17. Even though there is no label at node *3* that supports the binary relation *R(1, b, 2, b)*, label *b* should not be deleted from set A_2, since it may satisfy the constraints with other labels at node *1*. We should, instead, just delete the binary relation *R(1, b, 2, b)*, i.e., set *R(1, b, 2, b)* equal to false.

After algorithm PC-3 is finished, the final result will be:

$A_1 = \{\}$
$A_2 = \{c\}$
$A_3 = \{c\}$

This is not a correct result since from it we can not find the solution *(c, b, b)* of the constraint satisfaction problem. We will give a revised path consistency algorithm, PC-4, in the next section and analyze its time and space complexities.

3. Revised Algorithm PC-4

Figure 2 is the revised algorithm PC-4 for path consistency. In algorithm PC-4, *M* is a table with index *[i, b, j, c]* (in fact *M* is a boolean matrix). Set S_{ibjc} contains members of the form *(k, d)*, where the binary relations $R_{ik}(b,d)$ and $R_{kj}(d,b)$ are supported by the binary relation $R_{ij}(b,c)$. We also use counters with indices of the form *[(i, b, j,c), k]*. Counter*[(i, b, j, c), k]* is the number of admissible pairs *(i, b)-(k, d)* that support the binary relation $R_{ij}(b,c)$, where d is any admissible label at node *k*. Note that Counter*[(i, b, j, c), k]* is essentially always equal to Counter*[(j, c, i, b), k]*. If both of them equal to 0, then there is no admissible label at node *k* that can support the binary relation $R_{ij}(b,c)$. Hence, $R_{ij}(b,c)$ must be set to 0 (false).

```
Step 1

1 $M$:=0; $S_{ibjc}$:=Empty_set; List:=Empty; Counter:=0;
2 **for** $(i, j) \in E$ **do**
```

```
3  for $k = 1, n$ do
4    for $b \in A_i$ do
5      for $c \in A_j$ such that $R_{ij}(b, c)=1$ do
6        begin
7          Total := 0;
8          for $d \in A_k$ do
9            if $R_{ik}(b, d) = 1$ and $R_{kj}(d, c)=1$ then
10             begin
11               Total := Total + 1;
12               Append($S_{ibkd}$, $(j, c)$);
13               Append($S_{jckd}$, $(i, b)$);
14             end;
15         if Total = 0 then
16           begin
17             $M[i, b, j, c]$:=1; $M[j, c, i, b]$:=1;
18             $R_{ij}(b, c)$:=0; $R_{ji}(c, b)$:=0;
19           end else
20           begin
21             Counter[$(i, b, j, c), k$] := Total;
22             Counter[$(j, c, i, b), k$] := Total;
23           end;
24         end;
25 initialize List with $\{(i,b,j,c) | M[i,b,j,c]=M[j,c,i,b]=1 \text{ and } i<j\}$;
```

Step 2

```
26 while List not Empty do
27   begin
28     choose $(k, d, l, e)$ from List and remove it from List;
29     for $(j, c) \in S_{kdle}$ do
30       begin
31         Counter[$(k,d,j,c),l$]:=Counter[$(k,d,j,c),l$]-1;
32         Counter[$(j,c,k,d),l$]:=Counter[$(j,c,k,d),l$]-1;
33         remove $(j,c)$ from $S_{kdle}$;
34         remove $(k,d)$ from $S_{jcle}$;
35         if Counter[$(k, d, j, c), l$]=0 then
36           if $M[k, d, j, c]$=0 then
37             begin
38               $M[k, d, j, c]$:=1; $M[j, c, k, d]$:=1;
39               Append(List, $(k, d, j, c)$);
```

```
40          $R_{kj}(d, c) := 0$; $R_{jk}(c, d) := 0$;
41        end;
42    end;
43   for $(j, c) \in S_{lekd}$ do
44    begin
45      Counter[$(l,e,j,c),k$]:=Counter[$(l,e,j,c),k$]-1;
46      Counter[$(j,c,l,e),k$]:=Counter[$(j,c,l,e),k$]-1;
47      remove $(j,c)$ from $S_{lekd}$;
48      remove $(l,e)$ from $S_{jckd}$;
49      if Counter[$(l, e, j, c), k$]=0 then
50        if $M[l, e, j, c]$=0 then
51          begin
52            $M[l, e, j, c]$:=1; $M[j, c, l, e]$:=1;
53            Append(List, $(l, e, j, c)$);
54            $R_{lj}(e, c) := 0$; $R_{jl}(c, e) := 0$;
55          end;
56    end;
57  end
```

Figure 2. Algorithm PC-4

The complexity analysis of the algorithm PC-4 is similar to that of the algorithm PC-3. The maximum number of times line 11 to line 13 will be executed is on the order of n^3a^3 since $|E|$ is of order n^2 and $|A_i| = |A_j| = |A_k| \leq a$ (and k is from 1 to n in the for loop). For step 2, there are two ways to analyze its time complexity. First, since there are at most $O(n^3a^2)$ counters and each has a maximum value of a, line 31 to 34 and line 45 to 48 can be executed at most order $O(n^3a^3)$ times. (Remember that the index of Counter is of the form $[(i, b, j, c), k]$. There are $O(n^3)$ different i's, j's, and k's and $O(a^2)$ different b's and c's.) Second, the while loop is executed at most n^2a^2 times since each edge can be put into List at most once for each different pair of labelings of its two ends. The for loop is bounded by the size of S_{kdle} which is of order na. So the total time for step 2 is $O(n^3a^3)$. Therefore, the time complexity of the whole algorithm is $O(n^3a^3)$.

The space complexity of PC-4 is:

Number of counters $\leq O(n^3a^2)$,
Sum of the size of the different sets $S_{ibjc} \leq na \times \sum_{(i,j)\in N\times N} |A_i| \times |A_j| \leq n^3a^3$

(Since each set S_{ibjc} is of size na.)

So the space complexity of the whole algorithm is $O(n^3 a^3)$. Note that PC-4 has the same time and space complexity as PC-3.

4. Conclusion

Mohr and Henderson made an important observation which leads to the refinement of the arc and path consistency algorithms. However, they misuse labels at each node for the binary relations between a pair of nodes in the context of path consistency. In fact, one cannot delete the labels from the admissible labeling set of a node if there is any path inconsistency. Instead, one should remove the inconsistent relations by setting the relations to zero. This is the main error in the algorithm PC-3, and we have fixed it in the algorithm PC-4. Furthermore, the time complexity and space complexity of PC-4 are both $O(n^3 a^3)$.

5. Acknowledgements

The authors would like to thank the anonymous referees for their constructive remarks.

REFERENCES

1. R. Mohr and T.C. Henderson, Arc and Path Consistency Revisited, *Artificial Intelligence* **28** (1986) 225-233.
2. Alan K. Mackworth, Consistency in Networks of Relations, *Artificial Intelligence* **8** (1977) 99-118.
3. Alan K. Mackworth and Eugene C. Freuder, The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems, *Artificial Intelligence* **25** (1985) 65-74.
4. V. Montanari, Networks of Constraints: Fundamental Properties and Applications to Picture Processing, *Inform. Sci.* **7** (1974) 95-132.

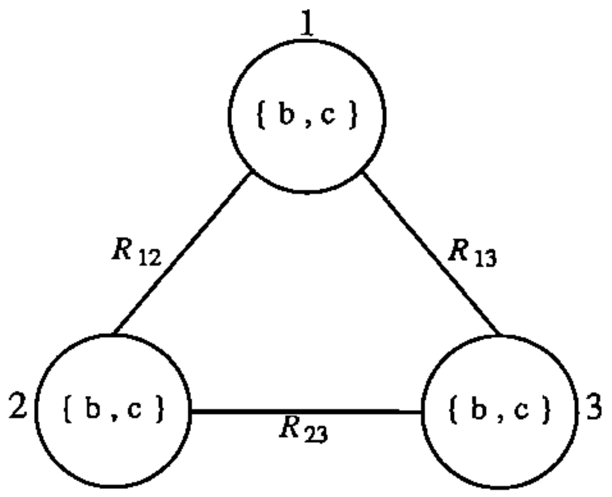

Figure 1. A counterexample to algorithm PC-3.