6F−3

制約充足問題の計算複雑さについて（2）

A Heuristic Backtrack-free Algorithm for Solving Constraint Satisfaction Problems

Jiang-Hong Li Seiichi Nishihara

University of Tsukuba

1.Introduction

In this paper, we develop a backtrack-free algorithm to solve constraint satisfaction problems[1], or CSPs for short. The algorithm, differing from the tree search, constructs a constraint tree with the constraint relations. We also show that a family of generalized class, we call it k-class, can offer increasing representational complexity for CSPs, while maintaining a bound on computational complexity linear in the number of variables and exponential in k by our algorithm.

Here, we restrict our attention to CSPs involving only binary constraints (in other word, |t| = 2). This restriction seems too critical, but it will be seen that the method to be presented can be applied to general CSPs without much modification. Another restriction is that there is not the same tuple in T, that is for any t_i and t_j($\in$T), if $i \neq j$ then $t_i \neq t_j$.

2. The Backtrack-free Algorithm

2.1 Background

The common algorithm for solving CSPs is the backtrack search algorithm. Backtrack builds partial solutions and extends them as long as they show promise to be part of a whole solution. When a dead end is recognized it backtracks to a preious variable. Clearly, if the next value can be guessed correctly, and if solution exists, the problem will be solved in time linear in the number of solutions with no backtracking. The device we wish to generate should order the candidates according to the confidence we have that they can be extended further to a full solution.

Such confidence can be obtained by constructing a constraint tree with the constraint relaions. And all solutions can be found by tracing back from the leaves of a constraint tree.

A constraint tree is generated by the following algorithm.

```
procedure CT-1
begin
 for i=1 to |T| do {
  (u,v)<-t_i;
  if u∈NODE
    insert (u,l) to tree;
       (for all l∈{l'|(l',l'')∈R_ti})
    NODE = NODE∪{u};
  else
    for all node (u:l) do
    if (there is not (l,l')∈R_i for any l'∈L)
       Delete (u:l) and its children;
   for all the tree which have root (u:l') do {
    if v∈NODE
      if (v:l'') is node, and not (l',l'')∈R_ti
         delete (v:l'');
    else
      {insert (v:l) to tree of roots (u:l');
         (for all l∈{l''|(l',l'')∈R_ti,})
         NODE = NODE∪{u};}
    }
 }
end
```

2.2 Upper bound complexity of algorithm CT-1

Assuming that m = |T|, d' = the average label number of 1 to u units. d = the average label number of units u'(u<=u',u'<=v for any (u,v) ∈ T). We have:

[Theorem 1] The computational complexity and space complexity of algorithm CT-1 are $\sum d'^{u}d^{v-u}$ and d'^{n}, which are exponential in n (n=|U|).

Proof For any (u,v)∈T, in order to find u, we will search a tree of level u, and will expend time d'^{u}. For every node (u:l) of the constraint tree, we will search a tree of level v-u to check whether the pair of (l,l') is in Ri and will expend time d^{v-u}.

Proving the space complexity is very simple, as a complete constraint tree has d'^{n} nodes.□

[Definition 1] An ordered CSP is one which satisfies:

Assuming T = $(t_1,\cdots,t_m)$, $t_i = (u_i,v_i)$, $t_j = (u_j,v_j)$,and i < j, we have

$u_i < u_j$ if $u_i \neq u_j$

$v_i < v_j$ if $u_i = u_j$

[Theorem 2] A CSP can be ordered in time $o(mn+m^2)$(Here, n=|U|,m=|T|)[3].

[Theorem 3] Let (U,L,T,R) be an ordered CSP. For any u∈U,if every (u',v)(∈T), where u'<u, has already been constructed, $(l_1,l_2,\cdots,l_{u-1})$ is always a pair of a final solution for (1,2,···,u-1).

2.3 The backtrack-free procedure

According to Theorem 3, it is possible to decrease the computational complexity of finding u by a procedure called merging, the merging procedure merges all subtrees which have the same root (u:l) (l∈L) to a tree, and

---

On the Computational Complexity of Constraint Satisfaction Problems: Part 2

Jiang-Hong Li. Seiichi Nishihara

University of Tsukuba

saves the solutions of (1,2,···,u-1) to all leaves of the tree. In other word, for any constraint tree, the leaves(u':l)(u<=u') always includes the solutions of (1,2,···,u-1).

The merging procedure is described as follows:

```
procedure CT-2
begin
 E={};
 NODE ={};
 (u,v)<-t1;
 u'=u;
 for (i=1; i<=|T|; i++) do {
   (u,v) <- ti;
   if (u'≠u)
     if for all l∈L, u∈NODE
        {insert node (u:l);
         NODE = NODE∪{u};}
     else
        {save root (u-1:l) to leaves.
         merge all subtrees of root (u:l);
         u'=u;}
     construct (u,v);
       /* the same as algorithm CT-1*/
 }
  merge all subtrees;
end
```

Fig 1 is an example which shows how to merge a constraint tree.

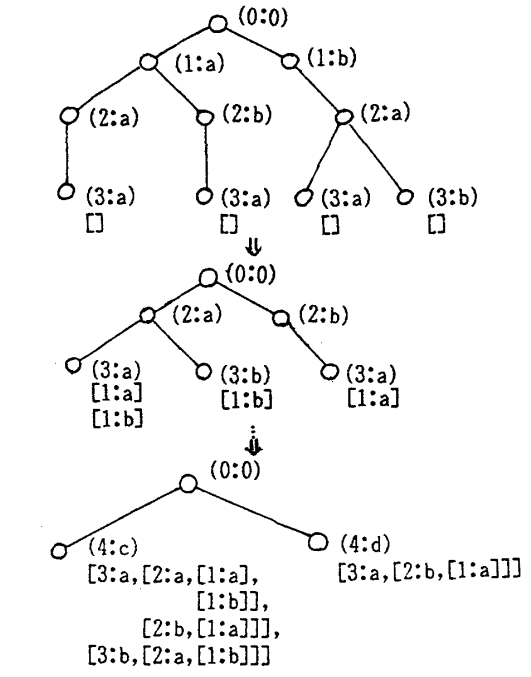

Fig.1 An example for solving a CSP by CT-2

The algorithm CT-2 decreases the computational complexity of constructing (u,v) from $\sum d'^{u} d^{v-u}$ to $\sum d' d^{v-u}$, but merging trees will spend time $O(d^h)$ (h is the height of the tree, in the worst case, h=n-1), and its space complexity is $O(s*d^h)$ (s is the size of solutions of 1,···,u-1) .For all h < |U|, the computational complexity of CT-2 is lower than CT-1's.

[Theorem 4] The algorithm CT-2 is backtrack-free.

3. Some properties of CLPs

[Definition 2] The width of an unit u is the number of v ∈{v|for all v > u, and u'<=u, (u',v) ∈ T}. the width of an ordering is the maximum width of all units, and the width of a CSP is the minimum width of all ordering of that CSP.

[Definition 3] The k-class of CSPs is a class of CSPs which have width k.

[Theorem 5] If a CSP is k-class, then its all solutions can be produced in time $O(d^{k+1})$.

It is because that all solutions of any CSP in k-class can be produced by CT-2 and the height of its constructed tree is k+1.

[Theorem 6] If a CSP is k-class, then its graph is a k-tree or a partial k-tree.(also see [5] about the definition of k-tree).

[Theorem 7] A CSP whose graph has the k-tree or partial k-tree structure is not always a k-class.

4. Conclusion

We have proved that there is a class called k-class whose computational complexity is bound on $O(nd^{k+1})$ for producing all solutions of CSPs by our algorithm, and that the set of graphs of k-class are a subset of partial k-tree. The mothod may be helpful even when the CSP is not k-class, it is also helpful for solving those CSPs which have negative constraint relations.

References

[1] Nishihara,S.,Li,J.: "On the Computatioal Complexity of Constraint Satisfaction Problems", Conf.IPSJ,42th,1991.

[2] Kruskal,C.P.,et,al: "Efficient Parallel Algorithms for Graph Problems", Algo. 5(1990) pp.43-64.

[3] Dechter,R.,Pearl,J.:"Network-Based Huristics for Constraint-Satisfaction Problems", Art.Int.34(1)(1988),pp.1-37.

[4] Freuder,E.C.: "Complexity of K-Tree Structured Constraint Satisfaction Problems", AAAI-90(1990),pp 4-9.